Sistemas Informáticos
Curso 2006-07

Estudio del rendimiento y la seguridad en redes ad hoc.

Alberto Benito Peral
Javier Sánchez-Moncayo Álvarez
Mario Morillo Molinuelo

Dirigido por:
Prof. Luis Javier García Villalba
Dpto. Ingeniería del Software e Inteligencia Artificial - LSI
Grupo de Análisis, Seguridad y Sistemas (GASS)

Facultad de Informática
Universidad Complutense de Madrid

Abstract:

Presently, wireless networks are achieving their great improvements due to the huge utilities they offer. In this document we have focussed special attention to MANET networks without an access point. We have realised a deep study into their characteristics, we have also made a strict and detailed classification of the different protocols existing up to now. Also, we have realised an intense study of some of these protocols, standing out how they work and how to implement them. Concretely, we have studied OLSR (Optimized Link State Routing Protocol) as an example of a proactive protocol. And for the reactive protocols we have studied DSR (Dynamic Source Protocol) and AODV (Ad Hoc On-Demand Distance Vector). Also, we have analysed how to integrate security in this last protocol protocol, called SAODV, including test and comparatives.

Resumen:

Actualmente, las redes inalámbricas están registrando sus mayores avances debido a la gran utilidad que ofrecen. En el presente documento nos hemos centrado en las redes MANET sin punto de acceso. Hemos realizado un estudio en profundidad de sus características, una estricta y detallada clasificación de los distintos protocolos que existen en la actualidad. Además, hemos realizado un intenso estudio de algunos de estos protocolos, dando detalles de funcionamiento y posible implementación. Más concretamente hemos estudiado el protocolo proactivo OLSR (Optimized Link State Routing Protocol) y varios protocolos reactivos, como son el DSR (Dynamic Source Routing) y el AODV (Ad Hoc On-Demand Distance Vector). Además, hemos hecho un análisis sobre la integración de la seguridad en éste último protocolo, denominado SAODV, dotado de pruebas y comparativas.

Palabras Clave:

MANET
Protocolo
OLSR
DSR
AODV
SAODV
Seguridad
Unicast
Multicast

Agradecimientos:

Gracias a los profesores Luis Javier García Villalba y Fabio Buiati por guiarnosen el desarrollo de este proyecto, así como al resto de personas que han colaborado en él. Gracias a nuestros familiares y amigos por apoyarnos durante este duro año.

INDICE

1. Introducción al proyecto

1.1. Introducción

Desde hace décadas las telecomunicaciones están en auge debido a las grandes ventajas y comodidades que ofrece. En este campo de la informática son indudables los grandes avances que se han ido produciendo a lo largo de los últimos años, y es inevitable pensar que aún no hemos llegado a una etapa final en dicha evolución. Aún estamos por ver grandes avances en este campo tan interesante. Más concretamente nos vamos a centrar en las redes inalámbricas, ya que, desde nuestro punto de vista, han entrado con mucha fuerza en todos los hogares principalmente a lo largo de los dos últimos años.

Las redes inalámbricas (WLAN) han sido uno de los principales motores del mercado de las telecomunicaciones en los últimos años. Con los últimos progresos de tecnologías VLSI aplicadas a microprocesadores y en tecnologías inalámbricas se ha reducido drásticamente tamaño y requisito de potencia de los sistemas. Cada vez se valora más la libertad de movimientos al no tener que estar “enchufado” a un cable, debido a la comodidad que ello conlleva. Es innegable vislumbrar un futuro prometedor para este tipo de redes.

Actualmente, en la mayoría de hogares y oficinas podemos llegar con nuestro ordenador y “mágicamente” conectarnos a una red local e incluso a Internet. Ahora bien, en nosotros no nos vamos a centrar exactamente en esté tipo de redes tan habituales, la particularidad de las redes que hemos estudiado es que no hay ningún punto de acceso donde nuestro ordenador vaya a conectarse, es decir, un punto de acceso, si no que cada ordenador se va a conectar directamente a otros ordenadores, formando así la red punto a punto. Las aplicaciones que este tipo de redes va a suponer las vamos a comentar más delante de forma más detallada, pero ahora no podemos dejar de imaginar la comodidad y la utilidad que puede suponer que una operación en el que trabajen distintos profesionales se conecten directamente entre sí y compartan información directamente entre ellos. Sin necesidad de ninguna infraestructura, sin necesidad de un ordenador central o punto de acceso que prepare y guíe las conexiones.

Este tipo de redes, conocidas como MANET son en las que hemos centrado nuestras investigaciones. A lo largo de la presente memoria, comentaremos las características más importantes de estas redes, presentamos estudios sobre distintos protocolos, así como pruebas realizadas sobre los mismos, además de un añadido sobre la seguridad en este tipo de redes y variantes propuestas en un protocolo escogido.

1.2. Objetivos

Pretendemos diseñar un protocolo eficiente que optimice la productividad, el consumo de energía y retrasos sin sacrificar “justicia”, estabilidad ni calidad de servicio (QoS)

Con productividad entendemos que queremos optimizar la fracción del ancho de banda de dedicado exclusivamente a transmisiones de datos, como ya hemos comentado, nos interesa trabajar con cabeceras de mensajes pequeñas. Cuando en un protocolo hablamos de "justicia" nos referimos al hecho de que durante las distintas comunicaciones que se producen en una red no exista ningún nodo con más prioridad que otros a la hora de retransmitir, y nos referimos también al hecho de que a ningún nodo, sea por el motivo que sea, se le impida o retrase indefinidamente su retransmisión de los paquetes hacia otro nodo.

También hemos definido como uno de nuestros objetivos que el protocolo sea estable y robusto Queremos evitar la perdida de paquetes en la medida de lo posible, pero hemos de admitir una posible perdida de los mismos. En particular tenemos que analizar los casos en los que el paquete perdido no es un paquete de datos si no un paquete de control (y que por tanto lleva información de control del funcionamiento del protocolo) Un buen protocolo debe funcionar aún con una perdida de un porcentaje de estos paquetes.

Finalmente, definimos la Calidad de Servicio, QoS, como el nivel de funcionamiento de un servicio ofrecido por la red a un usuario. Se puede "medir" mediante una serie de aspectos, como por ejemplo, el tiempo de entrega de un paquete, o la tase de entrega de paquetes y "low jitter".

1.3. Fases del proyecto

Nuestra base en el campo de las MANET era bastante pequeña, es por ello que hemos tenido que realizar un trabajo incremental de investigación en dichas redes. Comenzando desde los principios más básicos de las mismas hasta adentrarnos profundamente en las entrañas mismas de los protocolos.

Guíados por D. Javier García Villalba y Fabio Buiati hemos podido adquirir de forma paulatina pero profunda los distintos conocimientos para finalizar el año realizando pruebas y análisis detallados sobre distintos protocolos. Por ello, nuestro prior acometido fue realizar un pequeño trabajo donde extraimos todas las características de las redes MANET. Posteriormente hicimos un estudio sobre las distintas clasificaciones de los protocolos y extraimos distintos ejemplos de cada una de los grupos en los que se puede clasificar un protocolo. Esto lo realizamos para escoger el más representativo de cada grupo y realizar un estudio muy profundo de su modo de funcionamiento.

Después de hacer pruebas con los distintos protocolos estudiados, escogimos el protocolo reactivo AODV como nuestro protocolo base a partir del cual realizaríamos un estudio sobre la posibilidad de añadir un módulo de seguridad así como estudiar su comportamiento. Este protocolo será conocido como SAODV.

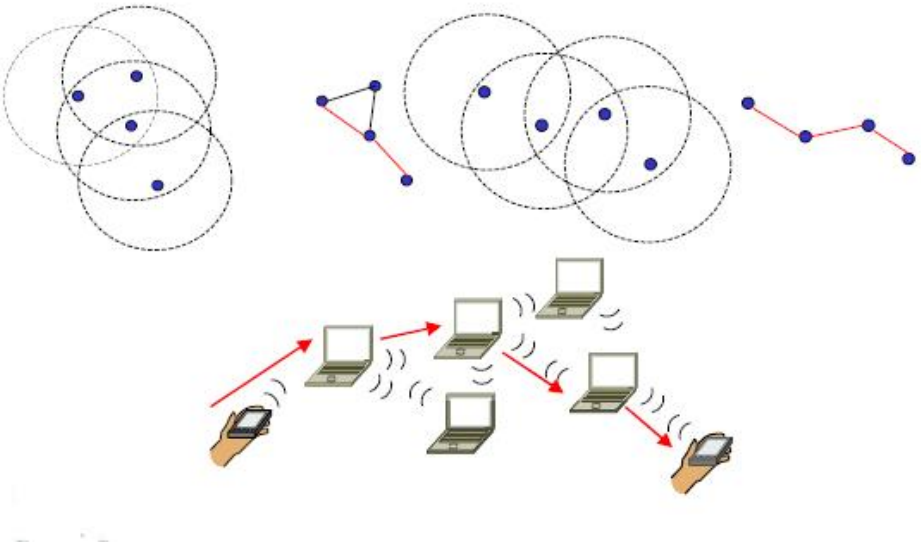
2. Redes ad hoc

Redes inalámbricas en las que no hay una infraestructura previa, sólo los propios terminales móviles, también conocidas como redes distribuidas, en el caso de tecnologías inalámbricas reciben el nombre de *MANETs* (*Mobile Ad Hoc Networks*)

Algunas de las aplicaciones principales de este tipo de redes son aquellas en las situaciones en las que no se dispone de una infraestructura previa, son, por tanto, adecuadas para reuniones, congresos, situaciones de emergencia, operaciones militares (FBCB2), redes de sensores (Smart Dust).

2.1. Visión general

Dentro de las redes inalámbricas establecemos 2 grandes grupos separados por una diferencia muy importante, la existencia de lo que conocemos comúnmente por un punto de acceso. Es decir, por un lado disponemos de una red con distintos nodos móviles que se pueden conectar a un nodo central o punto fijo el cual va a realizar funciones más importantes que el resto de nodos. Por el contrario, en el otro grupo no existe dicho punto de acceso, son las redes conocidas como Ad-Hoc.



Por tanto, mientras, en el primer grupo, la comunicación se puede realizar a través de dicho punto de acceso, la comunicación en el segundo grupo se realiza de forma punto a punto, es decir, de un nodo móvil a otro nodo móvil (nodo). De esta forma, en las Ad Hoc cada nodo tiene que actuar tanto de emisor como de receptor para que la comunicación entre dos nodos a través de la red se pueda realizar sin ningún tipo de problema.

Hemos de tener en cuanta también que estas redes se forman cuando se necesitan, que no requieren ninguna infraestructura previa y que todos sus

enlaces son inalámbricos. Por supuesto, al ser nodos móviles, no podemos olvidar en ningún momento que son aparatos cuya alimentación viene dada por el uso de baterías, en ningún momento podremos suponer que el nodo está conectado a la red. También, aprovechamos para mencionar, que salvo que se diga lo contrario, todos los nodos están dispuestos a cooperar unos con los otros, es decir, todos están dispuestos a retransmitir, encaminar, producir y consumir mensajes en tanto en cuanto que pertenezcan a la red Ad Hoc y tengan batería suficiente para hacerlo.

2.2. Características, inconvenientes.

Ya hemos comentado que en las redes Ad Hoc los nodos que la forman están dotados de movilidad, esto implica que, generalmente, no exista una topología fija, más bien al contrario, partimos de la base que la topología va a ser cambiante, es decir, en cualquier momento un nodo se puede desplazar de una parte de la red a otra, cambiando, por tanto, los enlaces que disponía con unos nodos de la red por otros nuevos. Es la principal característica a tener en cuenta, ya que la complicación de un diseño de un protocolo eficiente se basa principalmente en la movilidad de los nodos.

También, hemos comentado que, al contrario que otro tipo de redes inalámbricas, estos nodos no se conectan a puntos de acceso, si no que se conectan directamente a otros nodos. Esto supone que cada nodo disponga de una alta conectividad, ya que puede estar conectado a uno o a varios nodos a la vez.

La retransmisión de datos y de paquetes de control se hace sobre un medio compartido empleando ondas electromagnéticas, éste tipo de ondas conllevan una disipación de la señal con la distancia recorrida. Hablar de la disipación de la señal es lo mismo que comentar que cada nodo dispone de un área de retransmisión limitada. Así mismo, existe la posibilidad de colisión de las ondas con distintos objetos del medio o, simplemente, con otras ondas (recordemos que es un medio compartido) Estas colisiones producen interferencias en la señal, las cuales hacen que la tasa de errores sea más elevada.

Respecto al empleo de un medio compartido tenemos que añadir dos características vitales de las redes Ad Hoc. La primera es que el ancho de banda del canal es limitado. Este punto es muy importante a la hora de diseñar un protocolo ya que el envío de paquetes de control que realice cada nodo no debe de ser elevado para no desaprovechar aún más el ancho de banda, reducido, si cabe, por las interferencias ya comentadas. Adelantamos, que el empleo de cabeceras de los paquetes reducidas, ayudan a aprovechar el ancho de banda del mismo. La segunda, es que al ser un medio compartido nodos ajenos a la red pueden tener acceso a la información y a los paquetes que circulan por el medio. El tema de la seguridad será tratado más ampliamente en capítulos posteriores a lo largo de esta memoria.

Otro inconveniente es que cada nodo solo puede emplear transmisión Half-Duplex, debido a que una transmisión se colapsa porque su propia emisión genera interferencias.

Como dijimos anteriormente, los nodos disponen de un sistema de alimentación limitado a través de las baterías, lo cual hace que los nodos puedan perder área de retransmisión de la señal, o desconectarse de la red por no tener energía suficiente para permanecer conectado a la misma. Es importante que las operaciones que realicen los nodos en el tratamiento de paquetes sean de poco consumo, para maximizar el ahorro de energía.

Todas estas características suponen que los algoritmos de encaminamiento convencionales para Internet o redes LAN no sirvan. Dicho de forma coloquial, protocolos del estilo de RIP, OSPF o BGP no están pensados para tanto stress. Además, hay que tener en cuenta la posibilidad de que existan nodos unidireccionales en una red. Un nodo B es unidireccional si desde un nodo A puedo enviarle un paquete y que B lo reciba correctamente, pero desde el nodo B no puedo mandar un paquete directamente hasta A. En caso de que el envío desde B hasta A se pudiera hacer hablaríamos de nodos bidireccionales.

Debido a la movilidad el enrutamiento de paquetes tiene que ser calculado de forma dinámica, ya que el desplazamiento de un nodo por la red implica cambios en la ruta por la que deben viajar los paquetes. De la misma forma, dicha movilidad puede implicar la partición de la red en redes distintas en cualquier instante de tiempo. Este último aspecto está relacionado más con protocolos de autoconfiguración más que con protocolos de enrutamiento.

2.3. Operación de las redes Ad hoc.

2.3.1. Capa Física

Los nodos pertenecientes a una red inalámbrica se comunican entre si utilizando como medio de transmisión el espacio libre, es decir, se comunican mediante canales de radiofrecuencia (RF). Los canales de radiofrecuencias presentan características que dificultad la calidad de servicio en redes inalámbricas, algunas de los efectos que sufre la señal cuando se transmite en un canal RF son: el efecto *doopler*, la atenuación de la señal, el desvanecimiento por multitrayecto, etc. A la fecha, lo más común, es que los nodos de las redes ad hoc cuenten con antenas omnidireccionales que les permite comunicarse con cualquier otro dispositivo dentro de su rango de cobertura, el uso de este tipo de antenas influye en la velocidad de transmisión de las redes ah doc. Cuando un nodo desea transmitir o recibir, los nodos vecinos deben de estar en silencio haciendo que la capacidad de la red no se ocupe al 100%. Una de las soluciones propuestas en la literatura para solventar el problema de nodos vecinos es el uso de antenas unidireccionales.
Los estándares IEEE 802.11x definen interfaces para canales de RF en las bandas de los 2.4GHz y de los 5GHz, siendo la primera la más extendida. Esta banda esta muy saturada debido a que también es utilizada por otro tipo de

dispositivos y redes como lo son, los hornos de microondas y las redes **Bluetooth.**

Todo lo anterior da como resultado que los canales de RF utilizados por las redes ad hoc son poco fiables.

2.3.2. Capa MAC

En general, existen básicamente dos categorías principales de protocolos de control de acceso al medio: los protocolos de acceso aleatorio, en los cuales los nodos compiten entre sí para ganar el acceso al medio de transmisión compartido, y los protocolos de acceso controlado, en los cuales un nodo maestro o de infraestructura decide cuál de los nodos puede acceder al medio de transmisión en cada momento. La falta de una infraestructura y la naturaleza *peer-to-peer* de las redes ad hoc hacen que los protocolos de acceso aleatorio sean la opción natural para el control del acceso al medio en redes ad hoc.

Algunos ejemplos de los protocolos MAC utilizados en las redes ad hoc son los siguientes: MACA (*Multiple Access with Collision Avoidance*), MACAW (*MACA with Acknowledgment*), MACA-BI (*MACA by Invitation*) y FAMA (*Floor Acquisition Multiple Access*). De entre las diversas propuestas, el Comité IEEE 802.11 eligió a CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*), una variante de MACA, como la base para sus estándares, debido a su inherente flexibilidad y porque resuelve los problemas de los terminales oculto y expuesto a través del sencillo mecanismo de intercambio de señales de control RTS/CTSDATA/ACK.

Entre los Protocolos MAC de Acceso Controlado se encuentran los siguientes: TDMA (*Time Division Multiple Access*), FDMA (*Frequency Division Multiple Access*), CDMA (*Code Division Multiple Access*) y TSMA (*Time Spread Multiple Access*).

Aunque estos protocolos raramente se utilizan en redes ad hoc, se prefieren en ambientes que necesitan garantías de Calidad de Servicio (QoS), ya que sus transmisiones están libres de colisiones. Su aplicación está principalmente adaptada a redes como *Bluetooth* y a otras redes ad hoc basadas en grupos (*clusters*) de nodos, en las cuales el acceso al medio está controlado por nodos maestros que deciden qué nodo del grupo es el que tiene acceso al canal, posibilitando así transmisiones libres de contienda y colisión. Entre las propuestas para mejorar el comportamiento de los protocolos MAC para redes ad hoc basadas en protocolos de acceso aleatorio se incluyen, por ejemplo, algoritmos para reducir el consumo de energía de los nodos móviles que permiten que los nodos “duerman” (esto es, que pasen a un estado de bajo consumo de energía) durante el periodo ocioso; diversos algoritmos de gestión de paquetes; reducción de los radios de transmisión y de recepción; ajuste de la velocidad de transmisión entre nodos vecinos dependiendo de la distancia que los separa, y diferentes esquemas de codificación y modulación de señales.

2.3.3. Capa de Red.

Una característica especialmente importante de los protocolos de encaminamiento para redes ad hoc es que deben poder adaptarse rápidamente a lo cambios continuos de la red, con el fin de mantener las rutas entre los nodos que se están comunicando. De manera general, los protocolos de encaminamiento para redes ad hoc se clasifican en dos categorías principales: proactivos y reactivos, aunque también existen otras clasificaciones. Los protocolos proactivos mantienen tablas que almacenan la información de encaminamiento y periódicamente, o ante cualquier cambio en la topología de la red, disparan un mecanismo de propagación de actualización a través de la red, con el fin de mantener una idea real del estado de la red. Esto puede causar una cantidad importante de paquetes de señalización (*overhead*) que afecte la utilización del ancho de banda, el caudal (*throughput*), así como el consumo de energía. La ventaja es que las rutas a cada destino están siempre disponibles sin el incremento de paquetes de señalización que ocasiona un mecanismo de descubrimiento de rutas, pero tales protocolos tienen problemas para funcionar adecuadamente cuando en la red se presenta una alta tasa de movilidad o cuando hay un gran número de nodos en la red. Ejemplos de protocolos de esta categoría son: DSDV (*Destination-Sequenced Distance-Vector*), WRP (*Wireless Routing Protocol*), CGSR (*Cluster Gateway Switch Routing*), FSRP (*Fisheye State Routing Protocol*), OLSR (*Optimizad Link-State Routing*) y TBRPF (*Topology Dissemination Based on Reverse-Path Forwarding*). Los últimos dos protocolos preactivos se han convertido en RFC´s (*Requests For Comments*) experimentales aceptados por la IETF.

Por otro lado, los protocolos de encaminamiento por demanda o reactivos se caracterizan por iniciar un mecanismo de descubrimiento de ruta cuando una fuente necesita comunicarse con un destino al cual no sabe cómo llegar. El descubrimiento de ruta se realiza normalmente mediante una inundación de la petición. De manera general, el encaminamiento por demanda requiere menos *overhead* que el encaminamiento basado en tablas (proactivo), pero incurre en un retraso de descubrimiento de ruta cada vez que se requiere un nuevo camino.

Las diferencias entre los protocolos por demanda están en la implementación del mecanismo de descubrimiento de ruta y en las optimizaciones del mismo. Ejemplos de protocolos reactivos son los que se mencionan a continuación: El protocolo DSR (*Dynamic Source Routing*) que está muy cerca de convertirse en RFC experimental. Por su lado, el protocolo AODV (*Ad hoc On- Demand Distance Vector*) ya es un RFC experimental. Otro protocolo reactivo que inició el camino de convertirse en RFC pero que ya no ha tenido avances en ese proceso es TORA (*Temporally Ordered Routing Algorithm*). El protocolo de encaminamiento DYMO (*Dynamic MANET On-demand*) es el último protocolo reactivo unicast que está siendo considerado por el MANET-WG para convertirse en RFC, pero se encuentra apenas en sus primeras fases para lograrlo.

Además de los protocolos proactivos y reactivos están los protocolos híbridos. El protocolo ZRP (*Zone-Based Hierarchical Link State Routing Protocol*) es un ejemplo de protocolo híbrido que combina los enfoques proactivo y reactivo, tratando de combinar las ventajas de ambos.

2.3.4. Capa de transporte.

Las redes MANET tienen como objetivo final la implantación de la Internet inalámbrica omnipresente y ubicua, por lo que está basada en la pila de protocolos TCP/IP. Esto ha significado que prácticamente todos los esfuerzos de investigación y desarrollo del MANET-WG estén principalmente dirigidos al desarrollo de algoritmos de encaminamiento especialmente adecuados para este tipo de redes, sin que los demás protocolos de la pila se vean afectados. Aun así, en los últimos años se han presentado varias propuestas encaminadas a mejorar las prestaciones de los principales protocolos de transporte de la Internet, TCP (*Transmisión Control Protocol*) y UDP (*User Datagram Protocol*), en entornos ad hoc. Entre estas mejoras está el emplear mecanismos de señalización explícita que permitan tener un TCP "amigable" con las redes ad hoc, para que no active, erróneamente, el mecanismo de control de congestión ante la pérdida de la ruta entre el nodo fuente y el destino final debida a un fallo en el enlace. Otros autores han propuesto la creación de TPA (*Transport Protocol for Ad hoc Networks*), un nuevo protocolo de transporte especialmente diseñado para entornos ad hoc, que incluye mecanismos para detección de pérdida del enlace y recuperación de ruta, además de que establece un mecanismo de control de congestión diferente al de TCP. Los protocolos de transporte SCTP (*Stream Control Transfer Protocol*) y el MRTP (*Multiflow Realtime Transport Protocol*) han sido especialmente diseñados para ofrecer un mejor transporte de datos a las aplicaciones de tipo media-streaming y de tiempo real.

2.3.5. Capa de aplicación

Uno de los objetivos del MANET-WG es que las mismas aplicaciones diseñadas para la Internet actual puedan funcionar en las MANETs. Esto es válido actualmente para las aplicaciones de transmisión de datos que pueden conformarse con una entrega de datos bajo la filosofía de *Best Effort*, sin grandes requerimientos de ancho de banda ni restricciones en cuanto al tiempo de entrega de los paquetes. Sin embargo, Internet también se utiliza actualmente para otros tipos de aplicaciones, como aquellas que permiten el acceso en tiempo real a contenidos de audio y vídeo en repositorios remotos que producen flujos continuos de datos (*streams*), así como aplicaciones multimedia interactivas de tiempo real, como la voz sobre IP (*Video over Internet Protocol, VoIP*) o telefonía IP (*IP Telephony*), la vídeo conferencia y la vídeo telefonía, que demandan a la red de transporte ciertas garantías mínimas de retardo en la entrega de paquetes, de variación de este retardo y de ancho de banda, pero que por otro lado pueden soportar cierto nivel de pérdida de paquetes, sin que por tanto la información se vuelva inútil.

Debido a que el ancho de banda y la fiabilidad de los enlaces de las redes ad hoc es mucho menor que en las redes cableadas, hay muchos retos

por resolver para que este tipo de aplicaciones puedan ser soportadas de manera adecuada en los entornos ad hoc. Algunas propuestas van en el sentido de disminuir lo más posible el ancho de banda requerido por las aplicaciones, las cuales se basan en nuevas técnicas de codificación y compresión de la información, sacrificando la calidad de la información para todos los nodos y condiciones de la red por igual. Otra dirección que se está considerando cada vez más, es hacer que las aplicaciones mismas se adapten a las condiciones dinámicas de las redes ad hoc y a las capacidades particulares de los nodos comunicantes en ambos extremos.

2.4. Aplicaciones.

Es fácil encontrar situaciones donde se ve la utilidad de las redes Ad-Hoc. Uno de los ejemplos más clásicos (aunque también discutido) es una reunión de trabajo: un grupo de personas con ordenadores portátiles o PDAs. Son de distintas empresas y por tanto sus direcciones son distintas. Tal vez en la sala haya acceso a Internet y puedan usar por ejemplo IP móvil, pero ¿para qué pasear sus datagramas por todas la ciudad o todo el país cuando están en la misma habitación? Sus equipos probablemente estén dotados de puertos de infrarrojos o bluetooth que les permitan formar una red para la ocasión. En algunos casos, simplemente no habrá infraestructuras de apoyo. Pensemos en poblaciones aisladas o de orografía difícil, situaciones de emergencia, desastres naturales donde las infraestructuras hayan desparecido, etc. Otro ejemplo son las denominadas PAN (Personal Area Networks) o piconets: redes formadas por los dispositivos de una persona, como su reloj, su agenda y su teléfono móvil. Una red así puede querer entrar en contacto con la red de otra persona que en ese momento esté próxima.

La capacidad de desplegarse inmediatamente y la no dependencia de un único punto de fallo hace a estas redes muy interesantes para el uso militar, de hecho uno de los orígenes de esta idea está en la agencia de proyectos de investigación avanzada para la defensa (DARPA, Defense Advanced Research Projects Agency) del ministerio de defensa de Estados Unidos.

El campo militar es posiblemente el más desarrollado actualmente, el ejército estadounidense ya dispone de un sistema basado en este tipo de redes, el FBCB2 (Force XXI Battle Command, Brigade-and-Below). Uno de sus objetivos es distinguir las fuerzas propias de las fuerzas del enemigo, ofreciendo a los soldados una visión del campo de batalla similar a la de un vídeo-juego. Los equipos de la generación inmediatamente anterior estaban basados en comunicaciones por satélite, con latencias de cinco minutos. En abril de 2003 el FBCB2 se utilizó en la segunda guerra del golfo, lo que supuso probablemente el primer uso bajo fuego real de una red Ad-Hoc. Otro motivo por el que una red Ad-Hoc puede ser ventajosa es el coste. Aunque exista una infraestructura de red, si pertenece a una entidad ajena es muy posible que nos cobre por su uso, mientras que si tenemos nuestros equipos desplegados dispondremos ya de una red sin coste adicional. Por ejemplo los coches que pasan por una autopista podrían formar fácilmente una red Ad- Hoc, independiente de su capacidad de conectarse a otras redes como GSM, UMTS

o similar. Por último, supongamos que tenemos estaciones capaces de comunicarse empleando un satélite. Estos equipos de comunicaciones son caros, pero bastaría con que algunos tengan capacidad de conectarse al satélite para que todos dispusieran de conectividad. Y no todos los capaces de conectarse al satélite necesitarían estar conectados simultáneamente.

Además de las aplicaciones para redes Ad-Hoc de ordenadores que hemos mencionado, podemos citar un gran número de ámbitos especialmente indicados para el uso de redes de sensores:

- Seguridad: en nuestro entorno hay muchos elementos que pueden llegar a ser muy peligrosos: escapes de gas, instalaciones eléctricas en mal estado, contaminación de agua o aire, etc. Que un sensor perciba esto suele ser sencillo, el factor determinante es poder comunicarlo de forma adecuada en el sitio adecuado y con un coste adecuado. Muchos sistemas de seguridad han demostrado ser poco eficaces, bien por ser demasiado complicados y caros como para aplicarse masivamente, bien por estar muy limitados por la dependencia de las baterías, por emplear diferentes tecnologías propietarias no compatibles, o por no estar bien preparados para entrar en una red de comunicaciones.

- Domótica: una de las aplicaciones más atractivas para este tipo de redes. Dispositivos heterogéneos de diferentes fabricantes podrán comunicarse entre sí, librando al usuario de tareas triviales. Las luces pueden atenuar su intensidad cuando se encienda el televisor o el televisor reducir el volumen cuando suene el teléfono. Cada persona puede tener su propio perfil, al que se adapte de forma automática la temperatura, la luz, la música, la televisión o el ordenador, tanto en casa como en la oficina.

- Defensa: como hemos comentado anteriormente, uno de los ámbitos donde se ve mayor posibilidad de uso de las redes Ad-Hoc en general y de las redes de sensores en particular, es el militar. En 2005 se presentó un prototipo que empleaba motas MICA2 con sensores de sonido de bajo precio para la detección de francotiradores. El sistema es capaz de localizar el origen de un disparo, con precisión de 1 metro y latencia de 2 segundos, con tal de que esté separado 0,4 segundos de un segundo disparo. Podemos encontrar otro ejemplo en la industria del armamento, actualmente en fase de desarrollo avanzado: un campo minado auto-regenerable. Se trata de una red Ad-Hoc donde cada nodo es una mina anti-tanque. Si el enemigo abre una brecha en el campo, las minas lo perciben y tienen la capacidad de desplazarse para volver a cerrar el campo.

- Aunque se espera que las redes de sensores puedan reemplazar a este tipo de armas, las minas se usan mucho porque es una forma eficaz de evitar el movimiento del enemigo en un área remota. Desplegarlas resulta muy barato, pero desmantelarlas es muy caro, además son especialmente crueles porque siguen activas durante décadas, sin distinguir entre amigos, enemigos y población civil. Como alternativa,

una red de sensores puede desplegarse sobre el terreno para detectar de forma precisa al enemigo, lo que permitirá su destrucción por diversos medios (misiles, aviación, control remoto, etc.)

- Monitorización de instalaciones: la lectura de los contadores de agua y electricidad puede hacerse mediante una red de sensores, habrá un nodo por vivienda de un edificio o un barrio sin necesidad de tender nuevos cables. Un nodo principal recopilará la información para enviarla a la compañía suministradora. De esta misma forma se pueden monitorizar cableado eléctrico o cañerías de agua o gas, sin necesidad de un cableado paralelo.

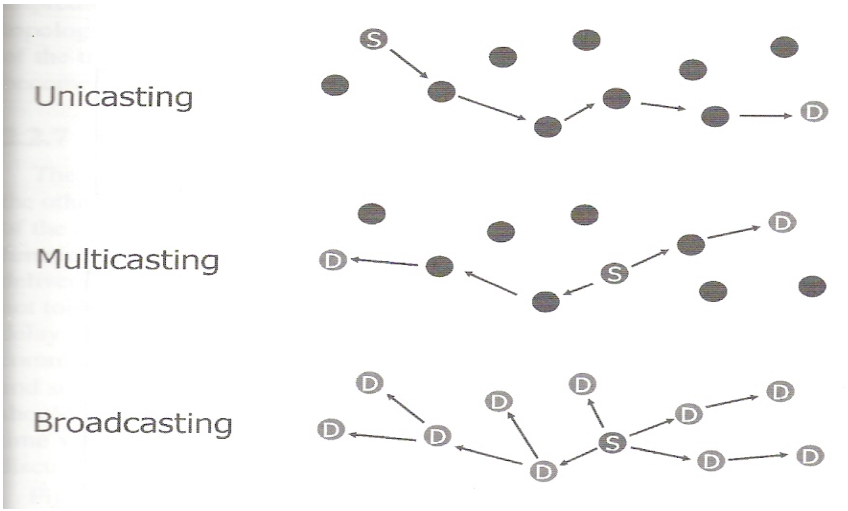
Podemos encontrar muchas otras aplicaciones, como el control de inventarios, de animales silvestres, monitorización del tiempo (lo que es especialmente útil en agricultura) y control de tráfico en carreteras para la detección de accidentes o atascos. En el ámbito de la salud pueden emplearse para monitorizar pacientes o para asistir a personas con discapacidad, en el de la seguridad, pueden detectar intrusiones. Son asimismo aplicables a diferentes sistemas de entretenimiento como juguetes o vídeo-juegos.

2.5. Enrutamiento

Por enrutamiento entendemos el conjunto de operaciones que realizan los distintos elementos que forman una red, desde el momento en el que un nodo origen desea enviar un paquete de datos a un nodo destino, hasta el momento en el que el mismo nodo origen ha recibido una confirmación de entrega de dicho paquete, o, por el contrario, hasta que el nodo origen desiste en el empeño de mandar dicho paquete, debido al fracaso de envíos iterativos y consecutivos del mismo.

Por ser una red Ad Hoc los elementos que forman dicha red van a ser los propios nodos de la red. El conjunto de las operaciones abarca operaciones de retransmisión de paquetes de control y de datos, cooperación con nodos vecinos, compartición de información de la red, así como distintas operaciones que según el protocolo pueden ser necesarias o no.

Recordemos, que en las redes *multi-hop* el nodo origen y el destino pueden estar separados por múltiples nodos, por ellos los datos deben ser reenviados a través de varios nodos hasta el destino. A continuación vamos a distinguir 3 tipos de enrutamiento en función del número de nodos a los que se de sea entregar un paquete. El primero, es el modo *"unicasting"*, donde un nodo origen quiere enviar un paquete a un solo nodo, destino. Es en este tipo de enrutamiento en el que se ha basado nuestro estudio, pero vamos a comentar también los otros dos tipos, así como formas de enrutamiento para ambos tipos. "*Multicasting"* es el enrutamiento en el que un nodo desea mandar un paquete a un conjunto de nodos destino, este conjunto, lo denominaremos "conjunto *multicast*". Finalmente, está el caso en el que un nodo desea enviar un paquete al resto de nodos de la red, este tipo de enrutamiento es conocido como "*broadcasting*".



2.5.1. Unicast

Existen varias formas de clasificar un protocolo. Éstas atienden a la forma de enrutamiento, al lugar de decisión del mismo, a una agrupación de los tipos de nodos de la red en cuanto a las funciones que van a desempeñar, o simplemente a características más generales.

2.5.1.1. Reactivos vs. Proactivos

En está clasificación se dividen los protocolos en función de la forma en la que se establece el enrutamiento. En los protocolos proactivos cada nodo dispone información lo más actualizada posible sobre la topología y estados de los enlaces de la red, de tal forma que cuando se desea enviar o retransmitir un paquete, el nodo encargado de ello sabe en dicho momento a que nodo tiene que enviarle dicho paquete. Esto proporciona una rápida respuesta ante solicitudes de ruta y ofrece un buen comportamiento en situaciones donde la tasa de movilidad es alta. Por el contrario, para disponer de información coherente y actualizada de la red es necesario que un nodo esté mandando cada cierto tiempo paquetes de control a sus vecinos para almacenar toda información pertinente. Por tanto, la ganancia del tiempo de respuesta se ve contrarrestada con la disminución del ancho de banda del canal y la memoria y procesamiento de toda la información que se capta en cada momento.

En los protocolos reactivos o bajo demanda los nodos no disponen información sobre la topología de la red, si no que crean las rutas cuando es necesario. La sobrecarga de almacenamiento y procesamiento es mucho menor que en los anteriores, pero los retrasos de establecimiento de la ruta son mucho mayores. A favor de estos protocolos añadiremos la existencia de caches de rutas, es decir, pequeñas memorias donde un nodo va a almacenar distintas rutas ya establecidas (por el o por compañeros suyos de la red) Así, con el uso de estas caches se puede mejorar la latencia de respuesta,

Finalmente en este apartado añadiremos la reciente aparición de unos protocolos híbridos en los que se mantiene una filosofía proactiva en un ámbito local, y reactiva en un nivel más global.

- Proactivos:
  - CGSR
  - DFR
  - DSDV
  - HSR
  - LCA
  - OLSR
  - TBRPF
  - WRP

- Reactivos:
  - Ad-hoc On-demand Distance Vector
  - Dynamic Source Routing
  - DYnamic Manet On-demand Routing
  - LBR
  - LMR
  - LUNAR
  - MOR
  - MPRDV
  - RDMAR
  - SSR
  - TORA

- Hibridos:
  - HSLS
  - ZRP

2.5.1.2. Jerárquico vs. Plano

En este caso la clasificación es mucho más sencilla. En los protocolos jerárquicos, los nodos pertenecen a diferentes niveles y su función en la retransmisión depende del nivel en el que esté. Normalmente, en el caso de este tipo de protocolos, las redes se dividen en grupos de nodos llamados *clusters.*

Por el contrario, en los protocolos planos, todos los nodos están al mismo nivel, y, por tanto, tienen las mismas funciones y responsabilidades

- CBRP (Cluster Based Routing Protocol)
- CEDAR (Core Extraction Distributed Ad hoc Routing)
- DART (Dynamic Address Routing)
- DDR (Distributed Dynamic Routing Algorithm)
- FSR (Fisheye State Routing protocol)
- GSR (Global State Routing protocol)
- HARP (Hybrid Ad Hoc Routing Protocol)

- HSR (Host Specific Routing protocol)
- HSR (Hierarchical State Routing)
- LANMAR (Landmark Routing Protocol for Large Scale Networks)
- OORP (OrderOne Routing Protocol)

2.5.1.3. Geográficos vs. No gegráficos

Como su propio nombre indica, en los protocolos geográficos se tienen en cuenta la posición geográfica exacta de cada nodo para realizar los encaminamientos. Su gran inconveniente es la necesidad de que cada nodo ha de incorporar un dispositivo de posicionamiento global (GPS). Principalmente, esto supone dos cosas: una, un aumento significativo del coste de cada nodo de la red, y dos, un aumento, también bastante significativo, del consumo de la energía.

Por el contrario, están los No geográficos que son aquellos protocolos que obvian la posición geográfica de cada nodo para establecer una ruta de comunicación entre los nodos.

- ALARM (Adaptive Location Aided Routing - Mines)
- BGR (Blind Geographic Routing)
- DREAM (Distance Routing Effect Algorithm for Mobility)
- GLS(Grid) (Geographic Location Service)
- LAR (Location-Aided Routing protocol)
- GPSAL (GPS Ant-Like Routing Algorithm)
- ZHLS (Zone-Based Hierarchical Link State Routing)
- GPSR (Greedy Perimeter Stateless Routing)

2.5.1.4. Encaminamiento en el origen vs Encaminamiento hop-by-hop

Tanto en redes convencionales como en redes Ad-Hoc, los protocolos pueden clasificarse atendiendo a dónde se guarda la información de encaminamiento. En el encaminamiento salto a salto (hop by hop routing) cada nodo decide solo la siguiente estación a la que enviará el paquete, atendiendo a sus propias tablas. Esto distribuye la complejidad por toda la red. En el encaminamiento en origen (source routing) la responsabilidad del encaminamiento recae sobre la estación origen del envío, que almacena en cada paquete la ruta completa que ha de seguir.

El encaminamiento en origen presenta una serie de ventajas claras:

1. Se garantiza la ausencia de bucles, evitando que nodos intermedios con información desfasada hagan que los paquetes sigan un camino errático.

2. Al nodo origen le resulta sencillo fijar rutas diferentes hacia un mismo destino sin necesidad de coordinarse con los nodos intermedios. Esto es útil para equilibrar la carga o para proporcionar diferentes calidades de servicio.

3. Cada paquete incluye la ruta completa que ha de seguir, esto permite que se propague valiosa información de encaminamiento por toda la red, sin coste adicional.

A estas ventajas se contrapone un único inconveniente principal: la información de encaminamiento ocupa mucho espacio en las cabeceras de los datagramas, reduciendo el espacio disponible para datos.

Los protocolos para redes convencionales que emplean salto a salto pueden dividirse a su vez en dos grupos: Estado del enlace, como OSPF (Open Shortest Path First). Cada nodo conoce el estado de toda la red y luego calcula la ruta óptima al destino aplicando el algoritmo de Dijkstra. En otros protocolos de estado del enlace no se distribuye la información completa sino en forma estadística, centrándose en las zonas que parecen más interesantes. Vector de Distancias (DV, Distance Vector), también conocidos como (DBF, Distributed Bellman Ford), donde cada nodo lo único que conoce de la ruta para llegar a otro es el primer salto y la distancia hasta el destino. Estos algoritmos presentan el inconveniente del salto al infinito: supongamos una ruta que pase por los nodos ABCD. Supongamos que B quiere encaminar un paquete hasta D, pero el enlace BC está roto (tiene peso infinito). Entonces B intentará encaminar hasta D pasando por A, porque A le ofrece un camino. Pero no puede saber que ese camino precisamente acabará pasando por el mismo enlace roto, circunstancia de la que A no tiene noticia aún. Algunas técnicas paliativas para este problema son las conocidas como split-horizon y poisoned-reverse. Todo lo que se conoce sobre protocolos de red convencionales se puede aplicar para redes Ad-Hoc, lo que supone una gran ventaja. En situaciones poco exigentes, y como primera aproximación, podríamos aplicar cualquiera de los protocolos convencionales de Internet en una red Ad-Hoc, ya que son capaces de funcionar sin configuración inicial (son self-starting), se adaptan a cambios en la topología y ofrecen múltiples rutas para un destino. Pero es necesaria una adaptación, ya que precisamente el peor comportamiento de los protocolos convencionales se da cuando los nodos se mueven con frecuencia, por ejemplo exigen aplicar continuamente el ya mencionado algoritmo de Dijkstra, que tiene complejidad exponencial.

En Internet, el encaminamiento se hace a partir de la direcciones. Históricamente se comenzó encaminando a partir de las clases, hoy se hace usando CIDR (Claseless Inter-Domain Routing). El principio es el mismo: a partir de cierto prefijo, de tamaño fijo o tamaño variable, se localiza el destino. Para las redes Ad-Hoc inicialmente se intentó una aproximación similar, pero resultó inadecuado por ser muy costoso: en redes de este tipo no hay relación entre una dirección de red y una ubicación física, por tanto no sirve de nada agregar prefijos en las direcciones y la escalabilidad es un problema serio: las tablas de enrutado corren el riesgo de hacerse inmanejables, al exigir una entrada por dirección y no por grupo de direcciones. Otro problema es el del tráfico de gestión de la red: una red donde los nodos se mueven, aparecen y desaparecen, puede generar muchos mensajes de control. Es importante encontrar un equilibrio: demasiados mensajes consumirán el ancho de banda

solo en mantener la red; un número muy bajo, hará la información de las tablas obsoleta.

Además, los mecanismos deben ser incrementales: sería muy ineficiente que una variación en un solo nodo obligase a recalcular la información de toda la red, puesto que los cambios son prácticamente continuos y las redes no tendrían apenas estados de estabilidad. Por todas las razones expuestas, se concluye que los protocolos de encaminamiento convencionales no son aplicables a las redes Ad-Hoc, que exigen algoritmos desarrollados específicamente para este entorno.

2.5.1.5. Multipath vs Singlepath

Esta es la clasificación más simple, y atiende simplemente al número de rutas que calcula y/o almacena un nodo. Un protocolo singlepath es aquen en el que solo se mantiene una ruta hacia cada destino. Por el contrario, los protocolos multipath son aquellos en los que se mantienen varias rutas hacia los distintos nodos destino.

2.5.2. Protocolos de Enrutamiento Multicasting

Como hemos dicho, en este caso uno o más nodos llevan información a varios nodos, por tanto, el objetivo del protocolo será obtener una calidad de Servicio (QoS) aceptable, así como minimizar la disipación de la energía y maximizar la eficacia del espacio reusado. Este último aspecto va relacionado con el nº de retransmisiones realizadas para entregar un paquete a sus destinos.

2.5.2.1. Flooding

Es la forma más simple de enrutamiento, en este caso un nodo manda el paquete a todos sus vecinos, y estos repiten el proceso, es decir, reenvían el paquete a sus vecinos. Se garantiza que el paquete llegará a todos los destinos deseados, a costa de usar mucho ancho de banda. Debido a que los paquetes lo van a recibir también algunos nodos que no pertencen al conjunto multicast se tiene que establecer la forma para que un nodo sepa si tiene que procesar el paquete (pertenece al conjunto) o por el contrario se tiene que limitar a reenviarlo (no pertenece al conjunto)

2.5.2.2. Aproximación de árbol

En este tipo de protocolos el nodo que desea enviar el paquete establece un árbol donde cada nodo del árbol va a ser un nodo de la red. La particularidad es que en la raíz se sitúa el nodo origen y en las hojas se sitúan los nodos pertenecientes al cojnjunto multicast. Los caminos desde la raíz a cada una de las hojas del árbol serán las rutas por las cuales deberán viajar los paquetes. Un ejemplo de este protocolo es el AMRIS.

2.5.2.3. Aproximación de malla (grafo)

En este caso, la topología de la red se plasma en un grafo, cada vértice del grafo identifica los nodos de la red, y las aristas representan los enlaces entre los distintos nodos. Éste tipo de enrutamiento se adapta mejor a las redes más dinámicas debido a la redundancia de esta aproximación. Permite que exista más de un camino entre origen y destino, pudiendo escoger alternativas en caso de que una de las probadas falle. El problema es que requiere una alta cantidad de intercambio de paquetes de control. Un ejemplo de este tipo de protocolos es el ODMRP.

2.5.2.4. Geographical multicast

Estos protocolos tienen las características vistas en los protocolos unicast geográficos, pero con las variaciones necesarias para adaptarlo a protocolos multicast.

2.5.2.5. Ejemplos

- **No geográficos**
  - ABAM (On-Demand Associativity-Based Multicast).
  - ADMR (Adaptive Demand-Driven Multicast Routing).
  - AMRIS (Ad hoc Multicast Routing protocol utilizing Increasing id-numberS).
  - MZR (Multicast Zone Routing).
  - ODMRP (On-Demand Multicast Routing Protocol).
  - SPBM (Scalable Position-Based Multicast).
  - SRMP (Source Routing-based Multicast Protocol).

- **Geographical Multicast (Geocast).**
  - LBM (Location Based Multicast).
  - GeoGRID (Geographical GRID (see GLS)).
  - GeoTORA (Geographical TORA (see TORA)).
  - MRGR (Mesh-Based Geocast Routing)
  - MOBICAST (Mobile Just-in-time Multicasting)
  - Abiding Geocast / Stored Geocast (Time Stable Geocasting).

2.5.3. Protocolos de Enrutamiento Broadcasting

Recordemos que las retransmisiones broadcasting un nodo quiere enviar información a todos los nodos restantes de la red a la que pertenece.

2.5.3.1. Pasivos parcialmente coordinados

Son algoritmos parecidos al de Retroceso Exponencial Binario. Un ejemplo sencillo es de CBB en el cual cada nodo lleva la cuenta de las veces que recibe el mismo paquete, así, si ese número es menor que un valor predeterminado lo reenvía, en caso contrario, lo deshecha. El protocolo DBB es otro ejemplo donde un nodo calcula la distancia del paquete mediante la potencia recibida del mismo. Ese nodo guarda todas las distancias de los

paquetes iguales, si la distancia más cercana es mayor que una distancia mínima predeterminada lo envía, en caso contraria, deshecha el paquete.

2.5.3.2. Activos parcialmente coordinados

En este caso cada nodo mantiene información de uno o dos saltos vecinos y/o información sobre la topología de la red. Una estrategia posible es que un nodo tome una decisión dependiendo de si el conjunto de nodos que guarda un vecino es el mismo. Para ello es necesario intercambiar información periódicamente con los vecinos. En esta categoría entran los algoritmos "clustering" que vamos a ver a continuación de forma más detallada.

2.5.3.3. Clustering

Esta solución es para llegar a un compromiso entre las dos vistas anteriormente. La solución centralizada reduce ampliamente el ancho de banda y no es escalable, mientras que la descentralizada sin coordinación mejora el ancho de banda pero aumentan las colisiones, sobre todo en las redes con una alta densidad de tráfico.

Es importante llegar a un compromiso entre ambas soluciones. El clustering permite obtener cierta coordinación entre los nodos sin llegar a saturar el ancho de banda de la rede con el envío de paquetes de control. En este tipo de algoritmos se escoge un "clusterhead" que actuará como un controlador local de los nodos de alrededor, es conjunto es un cluster. En una red puede haber varios clusterheads y un nodo puede ser miembro de distintos clusters. Estos protocolos requieren etapas de selección de clusterheads y etapas de mantenimiento de los clusters. Esto implica que sea nececesario, cada cierto tiempo, reducir el ancho de banda debido al envío de paquetes de control.

2.5.3.4. Ejemplos

- CBB
- DBB
- Highest-ID (clustering)
- Lowest-ID (Clustering)
- NTDR (Near Term Digital Radio, Clustering)
- HC (High Connectivity, Clustering)

2.6. Calidad de servicio (QoS).

El campo de las redes ad hoc aún se encuentra en los inicios, por lo cual todavía hay muchas áreas en las que se esta haciendo investigación. Uno de los problemas que no se han resulto satisfactoriamente es la calidad de servicio en este tipo de redes. En este capítulo se presenta un sondeo de las soluciones que se han propuesto a la fecha para soportar calidad de servicio en las redes ad hoc.

En estos tiempos existe una gran necesidad de superar la calidad de servicio ofrecida por *best effort* en redes móviles ad hoc. En contraste con las redes inalámbricas convencionales las redes ad hoc no posen una infraestructura ni una administración fijas. La interconexión entre nodos se lleva a cabo usando los terminales como *routers*, fuentes y destinos. Como los nodos en ah doc son dinámicos, es decir, se unen nodos a la red y otros la abandona, la topología de la red y las condiciones de tráfico también cambian dinámicamente haciendo extremadamente complicado dar buena calidad de servicio en aplicaciones multimedia. La limitación en ancho de banda y lo referente a la transmisión en el espacio libre y todas sus implicaciones también son razones que hay que tener en cuanta cuando hablamos de calidad de servicio en redes ad hoc. En la actualidad, lo que se hace en las redes con cables es sobredimensionarlas para poder ofrecer QoS, sin embargo, en las redes inalámbricas no podemos hacer eso debido a los escasos recursos con los que contamos.

2.6.1. Parámetros de QoS.

Las MANETs, y en general las redes basadas en IP, como la Internet, no ofrecen ninguna garantía en la entrega de los datos de extremo a extremo, sino que sólo ofrecen un servicio
Best-Effort. Sin embargo, los paquetes podrían no llegar a su destino por múltiples razones, como la congestión en alguna parte de la red, la pérdida de los enlaces entre los nodos, etc. Asimismo, las redes IP no pueden garantizar el tiempo de retardo de los paquetes ni un ancho de banda mínimo. Esto no representaba un problema cuando las redes IP se utilizaban para transmitir ficheros o correo electrónico en donde el tiempo de transmisión no era una restricción; TCP proporcionaba a la red las características necesarias para asegurar una entrega ordenada de todos los paquetes. Actualmente, sin embargo, están surgiendo nuevas aplicaciones para las redes IP que requieren de ciertas garantías mínimas de ancho de banda y tiempo de entrega de los paquetes de datos para poder funcionar adecuadamente. Por otra parte, a diferencia de la transmisión de ficheros o del correo electrónico, estas aplicaciones pueden soportar la pérdida de algunos paquetes de datos sin que a los usuarios les cause algún problema o siquiera lo perciban, siempre y cuando estas pérdidas sean esporádicas y mantengan tasas bajas. Así pues, las redes IP actuales deben soportar diferentes tipos de aplicaciones con diferentes requerimientos.

El término QoS se refiere a la garantía de recibir un servicio fiable de transmisión de información a través de una red. El objetivo de la QoS es proporcionar garantías sobre la capacidad de una red para entregar resultados predecibles. La QoS es de particular importancia para la transmisión continua de vídeo e información multimedia.

1) Parámetros de QoS. Los principales parámetros de QoS que se utilizan para medir las prestaciones de una red son la disponibilidad de la red, el ancho de banda, el retardo de los paquetes, la variación del retardo y la tasa de errores.

2) Modelos de QoS. Básicamente existen dos formas de proveer de QoS a una red IP. La primera consiste en sobredimensionar la red y la segunda en administrar adecuadamente los recursos con que cuenta mediante la aplicación de diversos métodos para gestionar los recursos, para que podamos ofrecer QoS en una red es necesario tener mecanismos como: control de admisión, función policía, etc. Obviamente, la primera es una solución muy costosa que requiere de actualizaciones continuas de todos los elementos de la red conforme crece la demanda de recursos. La segunda, por su parte, requiere de un esfuerzo mucho mayor.

Se han desarrollado dos modelos generales para proporcionar de QoS las redes IP: el modelo InstServ (*Integrated Services*) y el modelo DiffServ (*Differenctiated Services*). IntServ se basa en la reserva de los recursos necesarios para la transmisión de un flujo de datos desde la fuente hasta el destino.

Un usuario especifica los recursos que requiere mediante los parámetros de QoS y sus recursos son gestionados individualmente. IntServ requiere de un protocolo para la reserva y la gestión de los recursos, como RSVP (*Resources reSerVation Protocol*). Por su parte, DiffServ no requiere de la reserva de recursos por cada flujo de datos de extremo a extremo, sino que ofrece diferentes clases de servicio para soportar diferentes tipos de aplicaciones. DiffServ se vale del campo TOS (*Type of Service*) para el caso de IPv4, o del campo TC (*Traffic Class*) de IPv6 para indicar qué tipo de servicio debe recibir cada uno. Todos los paquetes de la misma clase reciben el mismo trato sin importar cuál es la fuente o el destino. IntServ puede ofrecer garantías estrictas de QoS para cada flujo admitido, pero tiene la desventaja de ser poco escalable debido a que mantiene variables de estado por cada flujo de la red. Por otro lado, DiffServ es fácilmente escalable, pero sólo puede asegurar un trato diferenciado por clases, por lo que no ofrece garantías estrictas de QoS.

2.6.2. QoS en redes Ad hoc.

Dotar de Calidad de Servicio a las MANETs es mucho más difícil debido a las características inherentes de estas redes. De manera general, los esfuerzos que se han desarrollado para proveer de QoS a la red Internet, y a otras arquitecturas de red basadas en infraestructura, no se pueden aplicar directamente en ambientes MANET. Para soportar QoS, se debe tener

información del estado de los enlaces, tal como retardo, ancho de banda, coste, tasa de pérdida y tasa de error en la red, y también se deben poder gestionar estas variables. Sin embargo, es muy difícil obtener y gestionar esta información en redes MANET, debido a la limitación de los recursos, a la movilidad y a la naturaleza aleatoria de entrada y salida de los nodos de la red.

Un Modelo de QoS especifica la arquitectura en la cual diversas clases de servicios puedan ser soportadas por las MANETs. Todos los otros componentes de QoS, tales como la señalización para la QoS, el encaminamiento con QoS, los mecanismos de control de admisión y los mecanismos para soporte de la QoS de la capa MAC, deben cooperar para lograr este objetivo. Así pues, uno de los principales retos consiste en definir un modelo adecuado para la provisión de QoS en redes ad hoc. Uno de los primeros modelos de QoS desarrollados específicamente para redes MANET es el modelo FQMM (*Flexible QoS Model for MANET*), el cual está basado tanto en IntServ como en Diffserv. Específicamente, para aplicaciones con alta prioridad, se proveen las garantías de QoS por flujo propias de IntServ. Por otro lado, las aplicaciones con menores prioridades obtienen una diferenciación en base a clases de servicios de DiffServ. Debido a que FQMM aplica ambos modelos de QoS de forma separada para aplicaciones con diferentes niveles de prioridad, las desventajas individuales de IntServ y DiffServ subsisten.

La provisión de QoS en las redes MANET es un verdadero reto. Sin embargo, es importante proveer a las MANETs de QoS para interconectarlas con las redes cableadas que soportan QoS (p. ej. ATM, Internet con QoS, etc.) y para soportar aplicaciones que requieran procesamiento en tiempo real.

Otra dirección para la provisión de QoS en redes ad hoc se basa en un modelo adaptativo de QoS: las aplicaciones deben poder adaptarse a las condiciones y a la disponibilidad de recursos de las redes ad hoc, que son altamente dinámicas. En la actualidad se están desarrollando muchos modelos para poder dotar con QoS a las redes ah doc. Estos modelos tienen en común ciertos elementos: control de admisión, control de congestión, función policía, algún mecanismo para diferenciar paquetes y algún mecanismo que nos permita conocer el estado de la red. En muchos de los modelos que se proponen se considera que las fuentes se adapten al estado de la red.

2.6.3. QoS en la Capa de Transporte.

TCP es un protocolo de transporte orientado a conexión que proporciona el control de flujo y control de congestión, esenciales para asegurar una entrega fiable de paquetes extremo a extremo. TCP fue originalmente diseñado para la transmisión de datos en redes fijas. Debido a que la tasa de error en las redes cableadas es bastante baja, TCP utiliza la pérdida de paquetes como una indicación de congestión en la red y corrige los problemas de manera efectiva haciendo los correspondientes ajustes a la transmisión en su ventana de congestión. Sin embargo en las MANETs son muchos los factores que impactan en el rendimiento de TCP. La movilidad puede causar fallos en los caminos y, por lo tanto, pérdidas de paquetes e incremento en los retardos. TCP malinterpreta estas pérdidas como congestión y pone en marcha el

mecanismo de control de congestión, lo que potencialmente conduce a retransmisiones innecesarias y degradación del caudal. Además, la movilidad de las estaciones puede magnificar la injusticia entre las sesiones TCP que compiten entre sí. En las redes ad hoc, aun cuando las estaciones sean estáticas, el rendimiento estará muy alejado del ideal debido a que la actividad de una estación está limitada por la actividad de las estaciones vecinas que se encuentran dentro del mismo rango de transmisión, así como por la interferencia causada por las estaciones ocultas y expuestas.

El tamaño de la ventana de congestión de TCP puede tener un impacto significativo en las prestaciones. En, los autores muestran que, para una topología de red y patrones de tráfico dados, existe un valor óptimo del tamaño de la ventana de congestión con el cual se maximiza la utilización del canal. Sin embargo, TCP no opera alrededor de este punto óptimo, sino que generalmente lo hace con una ventana mucho más grande, lo que conlleva a una reducción del caudal y a un incremento de la pérdida de paquetes. Estas pérdidas se deben principalmente a las caídas de la capa de enlace: una estación no consigue alcanzar su estación adyacente debido a la contienda e interferencia de otras estaciones.

Una pequeña ventana de congestión proporciona, en la mayoría de los casos, el mejor rendimiento. La interacción del protocolo MAC (IEEE 802.11x) con los mecanismos del protocolo TCP puede llevar a un comportamiento incierto en un ambiente multi-saltos. Estos fenómenos se pueden reducir/aumentar con el uso de pequeñas/grandes ventanas de congestión. Tales problemas no se presentan, o se presentan con menos intensidad, cuando se utiliza el protocolo UDP que, por otra parte, es un protocolo más adecuado para aplicaciones *mediastreaming* y de tiempo real.

Con el fin de resolver problemas específicos a las MANETs, se han propuesto muchos nuevos mecanismos de optimización de TCP, incluyendo la adaptación de las estrategias de detección de errores y de recuperación de TCP a ambientes de redes ad hoc. Para minimizar el impacto de la movilidad y la pérdida del enlace sobre las prestaciones de TCP, propuso introducir señalización específica (notificaciones de Fallo de Ruta y Reestablecimiento de Ruta) desde nodos intermedios, para notificar al TCP del nodo origen acerca de la pérdida del camino actual y la construcción de uno nuevo. De esta manera, TCP no activa los mecanismos de prevención de la congestión ante el fallo de un enlace, sino que simplemente congela su estado, el cual será retomado en cuanto se encuentre una nueva ruta. Asimismo, se ha propuesto el uso de un mecanismo de notificación específica de fallo de enlace, ELFN (*Explicit Link Failure Notification*). El objetivo de ELFN es proporcionar al TCP del origen indicaciones explícitas acerca del enlace y de los fallos en la ruta.

2.6.4. Señalización de QoS

La señalización de QoS es el proceso de establecimiento de una conexión desde un nodo fuente hasta un nodo destino que involucra la reserva de recursos en los nodos intermedios. La señalización de QoS actúa como un centro de control para el soporte de QoS. Reserva y libera recursos, establece,

termina y renegocia flujos en las redes. Los sistemas de señalización de QoS se pueden dividir en sistemas de señalización en banda o fuera-de-banda. En la señalización en-banda, la información de control se viaja dentro de los mismos paquetes de datos (*piggybacking*), mientras que en la señalización fuera-de-banda, la información de control se envía en paquetes explícitos.

INSIGNIA es un ejemplo de sistema de señalización en-banda para el soporte de QoS en redes ad hoc. Cuenta con algoritmos rápidos de reserva de recursos, restablecimiento de rutas y adaptación por flujo, los cuales están específicamente diseñados para proporcionar un servicio adaptativo en tiempo real en un ambiente de redes ad hoc móviles. Para establecer un flujo adaptativo en tiempo real, la información de señalización se transporta en cada paquete IP de datos, en el campo que se conoce como opción INSIGNIA. Cuando un nodo intermedio recibe un paquete con el valor apropiado en el campo de opción INSIGNIA, reserva los recursos si están disponibles y reenvía el paquete en dirección del nodo destino. El destino envía un mensaje de reporte de QoS a la fuente de forma periódica.

El reporte de QoS indicará a la fuente el estado de la red. Este reporte puede tomar una ruta diferente hacia la fuente. La fuente toma decisiones de adaptación con base en el reporte de QoS. Todos los nodos intermedios mantienen información de estado del enlace (*soft state*). La ausencia de tráfico producirá la recuperación o liberación de los recursos asignados al flujo para que puedan ser utilizados por otros flujos. Otros mecanismos para el transporte de señales de QoS en redes ad hoc son el SWAN (*Service Differentiation in Stateless Gíreles Ad hoc Networks*) y el Courtesy Piggybacking.

2.6.5. Capa de Red con Soporte para la QoS

El encaminamiento con QoS se refiere al descubrimiento y mantenimiento de rutas que pueden satisfacer objetivos de QoS bajo determinadas restricciones de recursos. Un protocolo de encaminamiento con QoS debe trabajar junto con la señalización de QoS para establecer caminos, a través de la red, que reúnan los requerimientos de QoS de extremo-a-extremo, tales como límites del retardo (*delay*) o de la variación del retardo (*jitter*), ancho de banda solicitado o restricciones multi-métricas. Una de las principales dificultades para los protocolos de encaminamiento con QoS en MANETs es que no se cumple el significado tradicional de que la QoS requerida debe ser asegurada una vez que se ha establecido un camino factible (*hard-QoS*). El recurso reservado puede no estar garantizado debido a la ruptura del camino, causado por la movilidad o porque se agote la energía de los nodos móviles.

El protocolo TBP (*Ticket-based Probing*) es un ejemplo de protocolo de encaminamiento con QoS. La idea básica del uso de *tickets* es limitar el número de caminos candidatos entre los que se realizará la búsqueda. Cuando una fuente quiere encontrar caminos con QoS hacia un destino, emite mensajes de prueba con algunos *tickets*. El número de *tickets* se basa en la información de estado disponible. Un *ticket* corresponde a la búsqueda de un camino y un mensaje de prueba debe transportar al menos un *ticket*. Así el número de *tickets* limita el número máximo de caminos buscados. Cuando un

nodo intermedio recibe un mensaje de prueba con n *tickets*, con base en su información local de estado, decide si divide los n *tickets* y cómo hacerlo, y hacia dónde reenviar los mensajes de prueba. Cuando el nodo destino recibe un mensaje de prueba, es que se ha encontrado un posible camino desde la fuente hasta el destino. Otro protocolo de red que ofrece soporte a la QoS es el QOLSR (QoS OLSR). Esta versión del protocolo proactivo OLSR puede ofrecer QoS para más de un parámetro de calidad (p. ej. ancho de banda y pérdida de paquetes). Entre otras técnicas que pretenden proporcionar un encaminamiento con QoS se encuentran la búsqueda preventiva de rutas alternas (*preemptive routing*) y el encaminamiento multi-camino (*multipath routing*).

2.6.6. Capa MAC con QoS

Un protocolo MAC con QoS debe resolver los problemas debidos a la contienda por el medio, los problemas de los terminales oculto y expuesto, debe soportar comunicaciones unicast fiables y debe proveer mecanismos de reserva de recursos para tráfico de tiempo real en ambientes inalámbricos distribuidos. Entre las numerosas propuestas de protocolos MAC que se han desarrollado para las redes inalámbricas, se encuentran algunas que ofrecen ciertas garantías de QoS para el tráfico de tiempo real en ambientes distribuidos como el protocolo GAMA-PS, el mecanismo de contienda *Black-Burst* (BB) y, más recientemente, el protocolo IEEE 802.11e.

2.6.7. Soporte de Capa Física a la QoS

La capa física de una red ad hoc también puede contribuir al soporte de la QoS, mediante el monitoreo de ciertos parámetros de transmisión y recepción de las señales, los cuales son realizados por los controladores de las tarjetas de red. A partir de dichos parámetros, se pueden obtener las métricas de QoS.

En general, una arquitectura de red por capas ha demostrado ser una manera adecuada de separar el problema de las comunicaciones de manera tal que cada capa realice ciertas funciones específicas de una manera más eficiente. Además, se tiene la ventaja de la flexibilidad, que implica que un protocolo de comunicaciones de una capa se puede sustituir por otro que realice las mismas funciones de diferente manera, generalmente más eficiente. Sin embargo, este modelo por capas presupone que cada capa puede ser optimizada de manera independiente y sin afectar las prestaciones del todo el sistema. Desgraciadamente esto último no se cumple en los entornos altamente dinámicos en los que constantemente se producen cambios en la calidad de los enlaces, la topología de la red y los requerimientos de QoS.

2.6.8. Conclusiones

Como se observa todavía no existe un modelo que satisfaga satisfactoriamente los requerimientos de calidad de servicio de las redes ah doc, es necesario seguir con la investigación para poder obtener un mecanismo que cumpla con las necesidades de las aplicaciones que requieren de calidad

de servicio y que se espera que puedan ser soportadas por las redes ad hoc. Proveer de QoS a éste tipo de redes ha sido un gran reto debido a las características de estas redes como la carencia de una estructura de control fija, la movilidad de los nodos, las complicaciones que se presentan en el canal de transmisión etc.

También se puede deducir que para poder dotar de QoS a las redes MANET es necesario tomar en cuenta todos los niveles de la red, ya que se requiere de la cooperación de éstos para lograr el objetivo.

2.7. Seguridad en redes Ad Hoc

2.7.1. Introducción

Las redes ad hoc inalámbricas son naturalmente vulnerables a diversos tipos de ataques y los protocolos de encaminamiento ad hoc no son una excepción. Existen diversas propuestas para proveer de seguridad a los protocolos de encaminamiento actuales. Una de ellas la podremos ver en la sección 3.3.4 con la seguridad incorporada por el protocolo SAODV.

En el contexto de las redes ad hoc debemos distinguir entre diferentes tipos de nodos que intentarán vulnerar la seguridad de estas redes:

- Un nodo *malicioso* será un atacante que no puede autenticarse a sí mismo como nodo legítimo debido a la falta de información criptográfica válida.

- Un nodo *comprometido* será un atacante interno quien se comporta de forma malintencionada, sin embargo puede ser autenticado por la red como un nodo legítimo y obtener la confianza de los otros nodos.

- Un nodo *selfish* (egoísta/acaparador) será un nodo que tendrá tendencia a denegar la provisión de servicios en beneficio de otros nodos con el fin de conservar sus propios recursos.

A partir de aquí podemos hablar de las necesidades y de los atributos de seguridad requeridos por un protocolo seguro y del protocolo seguro en sí.

2.7.2. Atributos de seguridad

En general, los principales atributos de seguridad que una red ad hoc debería cumplir son los siguientes:

- *Disponibilidad.*Significa que los servicios proporcionados por un nodo continúan siendo proporcionados con independencia de ataques. Los nodos deberían estar disponibles para la comunicación en todo momento, a pesar de sufrir ataques como denegación de servicios, los cuales pueden ser ejecutados en cualquier capa de una red ad hoc a través de radio jamming o agotamiento de batería, por ejemplo.

- *Autenticación.*Es esencialmente la confirmación de que las partes en una comunicación son auténticas y no son falsas, esto requiere que los nodos, de alguna forma, demuestren que sus identidades son las que ellos afirman ser. Sin autenticación un atacante podría hacerse pasar por un nodo, y así obtener acceso a información clasificada y sensible, y posiblemente causar interferencia con la normal y segura operación de la red.

- *Confidencialidad.* Asegura que un intruso no podría ser capaz de acceder a información en tránsito entre dos nodos. Se debe asegurar que la información no sea revelada a entidades no autorizadas. Es necesario prevenir que nodos intermedios y no confiables comprendan el contenido de los paquetes que están siendo transmitidos.

- *Integridad.* Es la garantía de que el mensaje o paquete que está siendo entregado no ha sido modificado durante el tránsito o de otra manera, y que lo que ha sido recibido es lo que originalmente se ha enviado. Un mensaje se podría corromper debido a razones no maliciosas tales como la debilitación de radio de propagación, pero existe siempre la posibilidad que un atacante haya modificado maliciosamente el contenido del mensaje.

- *No repudio.* Significa que el remitente de un mensaje no puede negar más adelante el envío de la información y que el receptor no puede negar la recepción. Esto puede ser útil mientras se detectan y aíslan nodos comprometidos. Cualquier nodo que reciba un mensaje erróneo puede acusar el remitente con la prueba y, así, convencer otros a nodos sobre el nodo comprometido.

- *Autorización.* Establece reglas que definen lo que cada nodo en la red tiene o no permitido hacer. En muchos casos se requieren determinar los recursos o información a la que el nodo puede acceder.

La seguridad en redes ad hoc asimismo tiene dimensiones adicionales tales como:

- *Cómputos livianos.* Muchos dispositivos conectados con una red ad hoc se asumen que son accionados con baterías con capacidades de cómputo limitadas. Por lo tanto, de tales nodos no se puede esperar que realicen cómputos costosos. Si operaciones tales como la criptografía de clave pública o algoritmos Shortest-Path para las grandes redes prueban que son necesarios tales cálculos, entonces dichas tareas deben ser asignadas al menor número posible de nodos - preferiblemente sólo a las extremos finales de la ruta en el momento de creación de ésta.

- *Anonimato*. Ni el nodo móvil ni su software de sistema deben por defecto exponer cualquier información que permita cualquier conclusión sobre el dueño o el usuario actual del nodo. En caso de que se utilicen identificadores de dispositivo o de la red (Dirección MAC o direcciones IP), ninguna asociación debe ser posible establecer entre el identificador respectivo y la identidad entre las partes de la comunicación.

- *Administración de Claves.* El servicio de administración de claves debe proveer alguna forma para responder a las siguientes preguntas:

- *Modelo de confianza* (Cuántos elementos diferentes en la red puede confiar mutuamente entre ellos y las relaciones de confianza entre ellos)
- *Criptosistemas* (Mientras el cifrado de clave pública es más conveniente, estos criptosistemas son significativamente más lentos que su contraparte de clave secreta cuando se necesita un nivel de seguridad similar)

- *Creación de la clave* (qué partes tienen permiso para generar claves para ellos mismos o para otras partes y qué clases de claves)

- *Almacenamiento de clave* (Cualquier elemento de la red puede tener que almacenar su propia clave así como posiblemente la clave de otros elementos)

- *Distribución de claves* (las claves generadas tienen que ser distribuidas de una forma segura a sus dueños)

- *Confianza.* Si la seguridad física es baja y las relaciones de confianza son dinámicas, entonces la probabilidad de una falla de seguridad puede incrementarse rápidamente. Construir seguridad por primera vez puede no ser tan difícil pero mantener la confianza y manejar los cambios dinámicos pueden requerir un mayor esfuerzo.

2.7.3. Ataques a redes Ad Hoc

Una vez enumeradas las características de seguridad deseables en una red ad hoc es necesario, ahora, distinguir los ataques que pueden sufrir estas redes. En general podemos diferenciar dos niveles de ataques:

- ataques a los mecanismos básicos de una red ad hoc como el encaminamiento.

- ataques sobre los mecanismos de seguridad y principalmente sobre los mecanismos de administración de claves.

Alternativamente los ataques, a su vez, pueden subdividirse en dos grupos:

- *Ataques pasivos*: típicamente involucran sólo la escucha (eavesdropping) de datos. Por ejemplo los ataques pasivos pueden incluir canales secretos, análisis de tráfico, sniffing de claves comprometidas, etc.

- *Ataques activos*: implican acciones ejecutadas por atacantes, como por ejemplo, replicación, modificación y eliminación de los datos intercambiados. Los atacantes activamente intentan modificar el comportamiento del protocolo en ataques de este tipo. La información es inadvertidamente revelada a los atacantes pasivos mediante los paquetes del protocolo que pueden ser usados para lanzar ataques activos.

Otra posible clasificación, desde el punto de vista del origen de los atacantes es la siguiente:

- *Ataques externos* son aquellos que están dirigidos a causar congestión, propagar información de encaminamiento falsa, evitar que los servicios funcionen apropiadamente, o detenerlos por completo. Estos ataques normalmente pueden ser evitados usando mecanismos de seguridad estándares como cortafuegos, cifrado, etc.

- *Ataques internos* son más severos que los externos, debido a que los nodos maliciosos internos ya pertenecen a la red como partes autorizadas y por lo tanto son protegidos por los mecanismos de seguridad que la red les ofrece. Así, tales nodos pueden hasta operar en grupo y usar los medios de seguridad estándares para proteger sus ataques.

2.7.4. Protocolos de encaminamiento seguros

2.7.4.1. Ataques al protocolo de encaminamiento

Existen varios tipos de ataques dirigidos directamente al protocolo de encaminamiento cuyo objetivo es interrumpir la operación de la red. Estos ataques pueden clasificarse de la siguiente manera:

- *Desbordamiento de la tabla de encaminamiento*: tiene lugar cuando un nodo atacante anuncia rutas hacia nodos no existentes a los nodos autorizados en la red. El objetivo es causar un desbordamiento de las tablas de encaminamiento, que a su vez evitaría la creación de entradas que correspondan a rutas nuevas a los nodos autorizados. Los protocolos de encaminamiento proactivos son los más vulnerables a estos ataques comparados con los protocolos de encaminamiento reactivos.

- *Envenenamiento de tabla de encaminamiento*: ocurre cuando los nodos comprometidos en las redes envían actualizaciones de rutas ficticias o modifican los paquetes de actualización de rutas genuinos enviados a otros nodos no comprometidos. Este tipo de ataque puede hacer que el encaminamiento deje de ser óptimo, puede congestionar porciones de la red, o hasta hacerlas inaccesibles.

- *Replicación de paquetes*: ocurre cuando un nodo atacante replica paquetes anteriores. Esto consume recursos de ancho de banda adicional y potencia de las baterías para los nodos y también causa un confusión innecesaria en el proceso de encaminamiento.

- *Envenenamiento de la caché de ruta*: en el caso de protocolos de encaminamiento reactivos (como AODV), cada nodo mantiene una caché de rutas la que almacena información recolectada de las rutas se

han hecho conocidas por el nodo en un pasado reciente. De la misma manera que en el envenenamiento de tabla de encaminamiento, un atacante puede también envenenar la caché de ruta para lograr objetivos similares.

- *Rushing Attack*: los protocolos de encaminamiento reactivos que usan eliminación de duplicados durante el proceso de descubrimiento de ruta son vulnerables a estos ataques. Un nodo atacante que recibe un paquete Route REQuest (RREQ) a partir de un nodo fuente inunda con el paquete rápidamente toda la red antes de que los otros nodos que reciben el mismo paquete RREQ puedan reaccionar. Los nodos que reciben los paquetes RREQ legítimos asumen que éstos son duplicados del paquete ya recibido proveniente del nodo atacante y por lo tanto los descartan. Cualquier ruta descubierta por el nodo fuente podría contener al nodo atacante como uno de los nodos intermedios. Por lo tanto la fuente no podría ser capaz de encontrar rutas seguras, es decir, rutas que no contengan al nodo atacante.

- *Black Hole*: ocurre cuando un nodo malicioso usa el protocolo de encaminamiento para anunciar que él posee la ruta más corta a los nodos de los paquetes que el quiere interceptar. En un protocolo basado en inundación o flooding, el atacante escucha las solicitudes de ruta, cuando recibe una solicitud para una ruta al nodo objetivo, el atacante crea una respuesta que consiste en una ruta extremadamente corta. Si la respuesta del nodo malicioso llega al nodo solicitante antes de que la respuesta del nodo actual, se crea una ruta falsa. Una vez que el nodo malicioso es capaz de insertarse entre la comunicación de los nodos, puede hacer cualquier cosa con los paquetes que se transmiten entre ellos, por ejemplo, elegir eliminar paquetes para realizar una denegación de servicios o usar su ubicación en la ruta como primer paso para un ataque main-in-the-middle.

- *Privación de sueño(sleep deprivation)*: este tipo de ataque es práctico sólo en aquella redes ad hoc donde la vida de las baterías es un parámetro crítico. Los dispositivos intentan conservar la energía de sus baterías sólo transmitiendo cuando es necesario. Un atacante puede intentar consumir baterías mediante la solicitud de rutas o mediante el reenvío innecesario de paquetes a un nodo, por ejemplo, mediante un ataque black hole.

- *Divulgación de ubicación*: este ataque consiste en revelar algún tipo de información acerca de la localización de los nodos o la estructura de la red.

Otros ataques de tipo multicapa y que también atentan contra los protocolos de encaminamiento son:

- *Denegación de Servicios*: un atacante procura evitar que los usuarios legítimos y autorizados de los servicios ofrecidos por la red tengan acceso a esos servicios. Este es un tipo de ataque multicapa, pero en

particular en la capa de red, un atacante podría tomar parte del proceso de encaminamiento y explotar el protocolo de encaminamiento para interrumpir el normal funcionamiento de la red.

- *Suplantación de identidad*: un atacante asume la identidad y privilegios de un nodo autorizado, con el fin de utilizar recursos de la red que no están disponibles bajo circunstancias normales o para interrumpir su normal funcionamiento mediante la inyección de información de encaminamiento falsa. Un atacante podría enmascararse como un nodo autorizado usando varios métodos, por ejemplo el ataque de man-in-the-middle.

2.7.4.2. Encaminamiento Seguro

La operación segura de un protocolo de encaminamiento de una red MANET es de principal importancia debido a la ausencia de una infraestructura fija. En su lugar, los nodos se asocian transitoriamente y cooperan virtualmente con cualquier otro nodo, incluyendo aquellos que podrían potencialmente interrumpir las operaciones de descubrimiento de ruta y reenvío de datos. En particular, la interrupción del descubrimiento de ruta puede ser un medio efectivo para obstaculizar sistemáticamente el flujo de datos. Los atacantes pueden responder con respuestas de rutas obsoletas o corruptas o difundir paquetes de control falsos para obstruir la propagación de consultas y actualizaciones de rutas legítimas.

Una vez vistos las propiedades de seguridad de una red ad hoc y los ataques que puede sufrir, particularmente contra la información de encaminamiento, los requisitos fundamentales que deberían tenerse en cuenta en el momento de diseñar un protocolo de encaminamiento seguro:

- *Detección de nodos maliciosos*: un protocolo de encaminamiento seguro debería ser capaz de detectar la presencia de nodos maliciosos en la red y debería evitar la participación de dichos nodos en el proceso de encaminamiento. Aún si tales nodos maliciosos participan en el proceso de descubrimiento de ruta, el protocolo debería elegir caminos que no incluyan a tales nodos.

- *Garantía de descubrimiento de ruta correcta*: si existe una ruta entre los nodos fuente y destino, el protocolo de encaminamiento debería ser capaz de encontrar la ruta y debería asegurar la exactitud de la ruta seleccionada.

- *Confidencialidad de la topología de red*: como ya vimos, un ataque de acceso a la información puede conducir al descubrimiento de la topología de la red por parte de nodos maliciosos. Una vez que la topología de la red es conocida, el atacante puede intentar estudiar el patrón de tráfico en la red. Si algunos de los nodos que son encontrados presentan más actividad que otros, el atacante puede intentar montar, por ejemplo, un ataque por denegación de servicio (DoS) sobre esos nodos que representan cuellos de botella. Esto puede afectar en última

instancia al proceso de encaminamiento en curso. Por lo tanto la confidencialidad de la topología de la red es un requerimiento importante a ser cumplido por los protocolos de encaminamiento seguros.

- *Estabilidad contra ataques*: el protocolo de encaminamiento debería ser auto-estable en el sentido de que debería poder volver a su estado de operación normal dentro de límites de tiempo razonables luego de haber sufrido un ataque pasivo o activo. El protocolo debería tener en cuenta que estos ataques no interrumpan permanentemente el proceso y debería asegurar robustez frente a ataques llamados de tipo Binzantino, esto es, que el protocolo debería trabajar apropiadamente aún si algunos de los nodos, los cuales anteriormente habían participado en el proceso de encaminamiento, ahora se convierten en nodos maliciosos o están intencionalmente deteriorados.

En una red ad hoc, desde el punto de vista de un protocolo de encaminamiento, se diferencian dos clases de mensajes: los **mensajes de encaminamiento** y los **mensajes de datos**. Ambos tienen distintas necesidades de seguridad. Los mensajes de datos son punto-a-punto y pueden ser protegidos con sistema de seguridad punto-a-punto (como IPSec). Por otro lado, los mensajes de encaminamiento son enviados a vecinos intermedios, procesados, posiblemente modificados, y reenviados. Más aún, como resultado del procesamiento del mensaje de encaminamiento, un nodo puede modificar su tabla de encaminamiento. Esto genera la necesidad de que los nodos intermedios sean capaces de autenticar la información contenida en los mensajes de encaminamiento (necesidad que no existe en las comunicaciones punto-a-punto) para poder aplicar sus políticas de seguridad.

Otra consecuencia de la naturaleza de la transmisión de mensajes de encaminamiento es que, en muchos casos, existirán partes del mensaje que se modifiquen durante su propagación. Esto es muy común en los protocolos por Vector de Distancia, donde los mensajes de encaminamiento usualmente contienen un contador de salto de ruta que ellos solicitan o proveen. Por lo tanto, en un mensaje de encaminamiento se podrían distinguir entre dos tipos de información: **información mutable** e **información no mutable**. Se desea que la información mutable en un mensaje de encaminamiento sea segura de tal forma que no necesariamente exista confianza entre los nodos intermedios, aunque, asegurar esta información es mucho más costoso en cómputo.

2.7.5. Esquemas de seguridad.

2.7.5.1. Esquemas de seguridad cooperativa.

Los esquemas de seguridad cooperativa para redes ad hoc se pueden clasificar en tres grupos:

- Basados en monedas virtuales, en el que los usuarios reciben un incentivo para cooperar en las operaciones de la red. Ejemplos: Nuglets, Cashnet

- Basados en monitorizado local, de forma que cada nodo monitoriza a sus vecinos evaluando para cada uno una métrica que refleja su comportamiento (reputación) hasta ese momento, lo que permite aislar gradualmente a los nodos egoístas. Ejemplo: Confidant, CORE.

- Híbridos: combinan ambos conceptos, ya que cada nodo debe poseer un vale para participar en la red, y son sus vecinos, que monitorizan su comportamiento, quienes deciden al vencimiento de dicho vale, si puede renovarlo o no.

Presentamos los siguientes ejemplos de esquemas de seguridad cooperativa:

- Nuglets: basado en el pago por salto (hop) en el envío realizado de cada paquete, mediante un contador en un módulo seguro en cada nodo, para animar a la cooperación en los envíos.

- Confidant: detecta y aísla los nodos egoístas, restando atractivo a la no cooperación.

Nuglets	Confidant
Se limitan a interacciones uno-a-uno	Un mal comportamiento produce una mala reputación que se propaga a más de un nodo
Requiere un módulo de seguridad a prueba de falsificaciones	No requiere módulo de seguridad a prueba de falsificaciones

2.7.5.1.1. Nuglets.

Tiene dos objetivos. Los usuarios deben ser:

- Incentivados para contribuir a las operaciones de la red (especialmente a la distribución de paquetes de otros nodos).

- Desanimados a sobrecargar la red.

La solución aportada por este esquema de basa en monedas virtuales llamadas “nuglets”. Existe un problema derivado de la localización dentro d ela red tienen muy diferentes oportunidades para ganar monedas, lo que no es justo para todos los nodos. Normalmente los de la periferia tienen menos oportunidades para ser recompensados. Se distinguen dos modelos:

- Monedero de paquetes (Packet pursel model, PPM). Cada paquete es cargado con nuglets por la fuente, y cada nodo intermedio de la ruta cobre nuglets por servicio realizado. Paga la fuente.

- Comercio de paquetes (Packet trade Model, PTM). Cada paquete es comprado con nuglets por cada nodo intemedio al nodo previo en el

camino, de forma que la diferencia entre lo que se cobre y lo que se paga es la ganancia. Paga el destino.

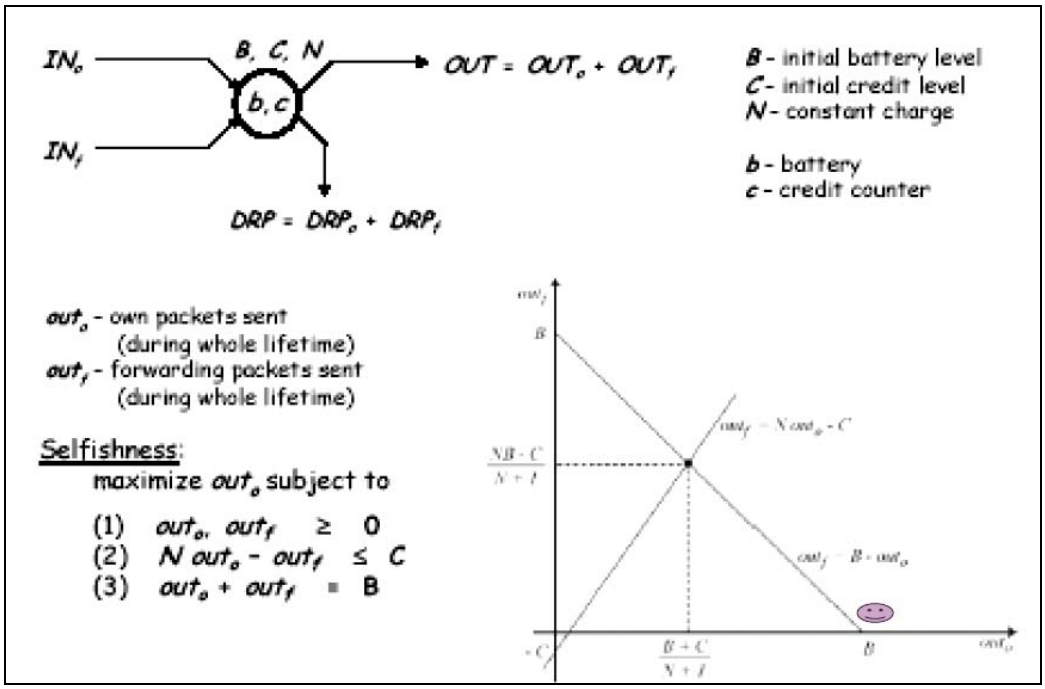
	Ventaja	Desventaja
PPM	Desanima a la sobrecarga de la red	La fuente necesita saber cuántos nuglets tiene que incluir en el paquete que envía.
PTM	La fuente no necesita saber cuántos nuglets necesita un paquete.	No se puede evitar la sobrecarga maliciosa de la red.

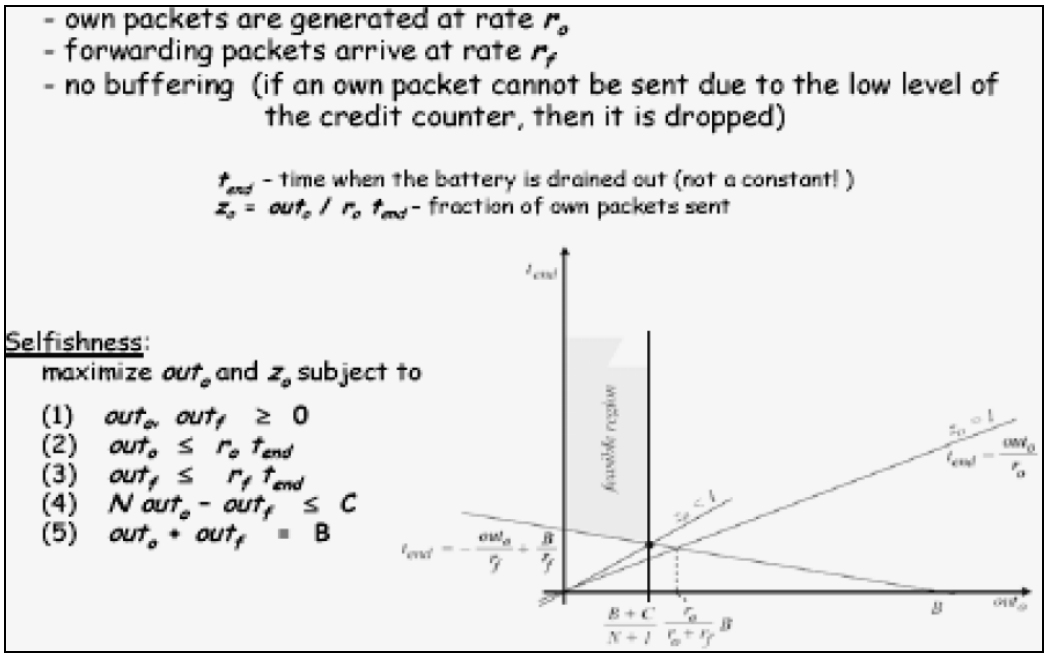
Una alternativa es el modelo híbrido con nuglets:

- Cada paquete se maneja según el PPM hasta que se agotan los nuglets cargados. Entonces se maneja según el PTM hasta que el receptor lo compra.
- Desanima a la sobrecarga de la red.
- La fuente no necesita saber cuánto nuglets necesita el paquete.

Extensiones del PPM

- Con cargos fijos por salto.
- Con pujas en cada salto. Implica una sobrecarga en las comunicaciones por el necesario intercambio de mensajes entre vecinos. Además, requiere que el algoritmo de enrutamiento permita a cada nodo tener múltiples entradas para diferentes caminos hacia el mismo destino. La ventaja es que puede suponer un ahorro de nuglets.





2.7.5.1.1.1. Simulación.

Hipótesis de la simulación:

- La fuente de un paquete conoce la posición del destino, la propia y la de sus vecinos.

- Antes del envío, pone las coordenadas, determina cuál es el vecino más cercano al destino.

- Cada nodo intermedio hace lo mismo, y si no se tiene ningún vecino más cercano al destin que él mismo deja perder el paquete.

- En la extensión de cargos fijos, el nº de nuglets que la fuente necesitaría para enviar un paquete a la distancia máxima en la red.

El escenario estaba constituido por:

- 400 nodos inmóviles, colocados al azar en un toro de 1000 m. x 1000 m. (para evitar efecto borde).

- Cada nodo tiene el mismo alcance de 100 m. y genera 0,5 paquetes/seg.

- El destino se escoge al azar.

- Los paquetes son de 4000 bits.

- El consumo de batería y los valores iniciales de nuglets también se analizan.

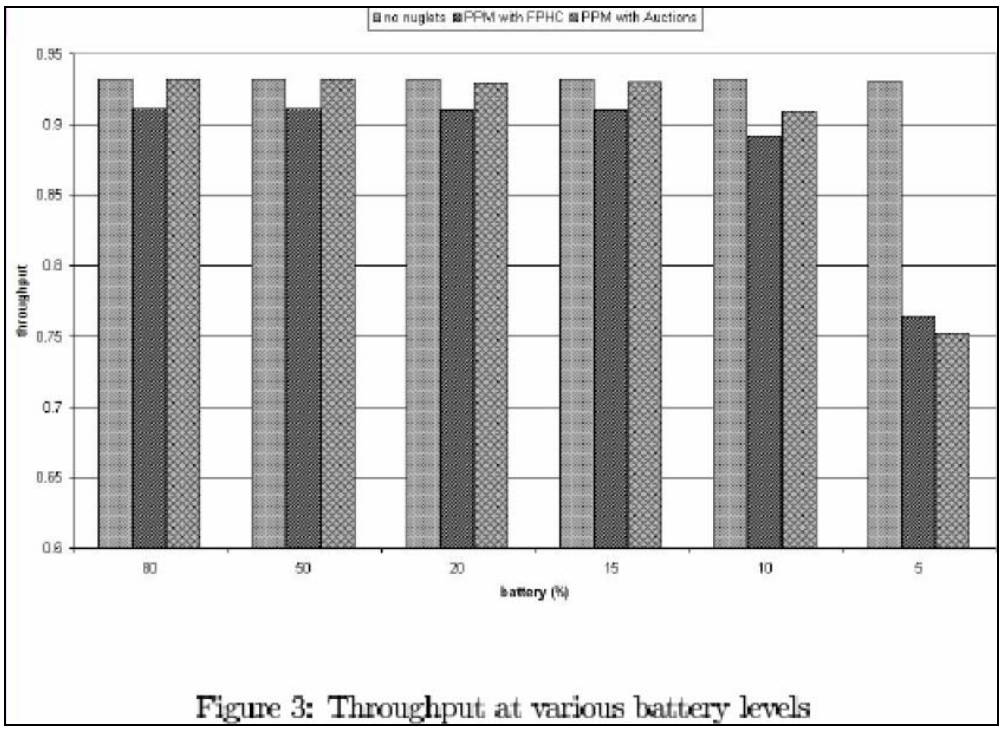
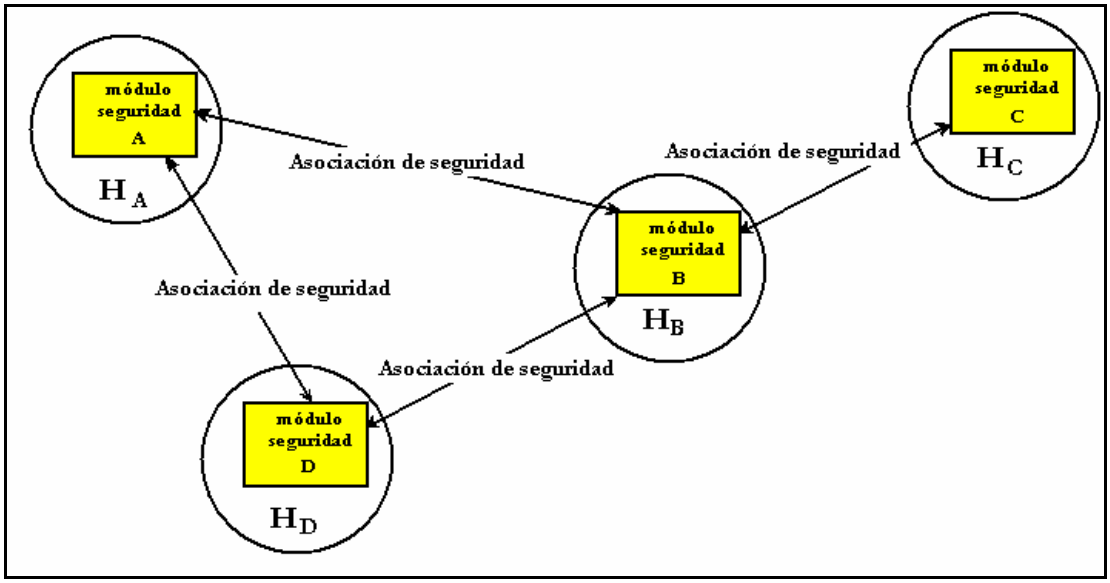


Figure 3: Throughput at various battery levels

2.7.5.1.1.2. Protocolo de hello.

Asumimos tres condiciones fundamentales:

- Se supone que los vecinos no cambian mucho.
- A intervalos regulares se ejecuta el protocolo hello para descubrir vecinos y establecer asociaciones de seguridad mediante claves de sesión compartidas k_{AB}.
- Se usa un esquema típico de PKC para el intercambio de claves de sesión secretas compartidas.



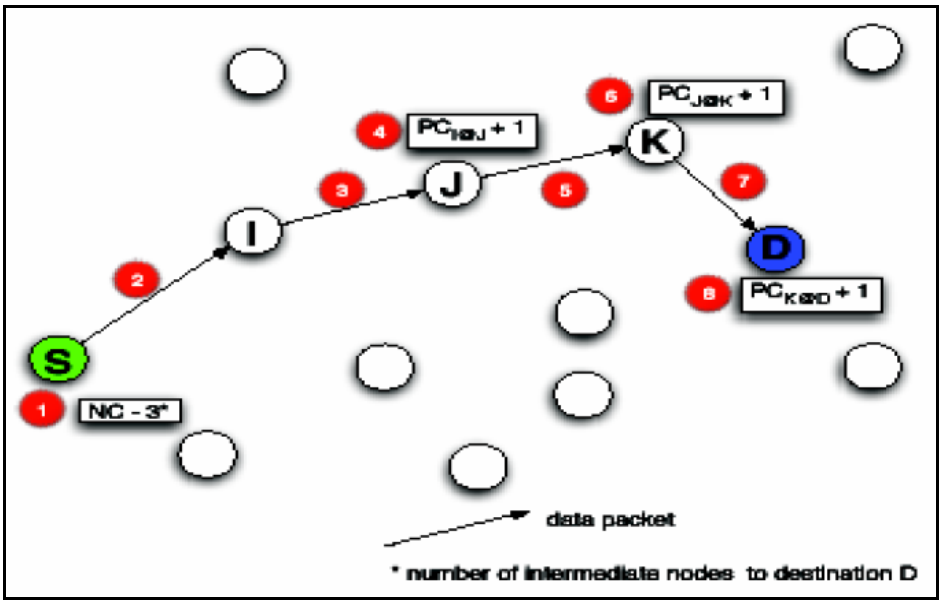
2.7.5.1.1.3. Módulo de seguridad.

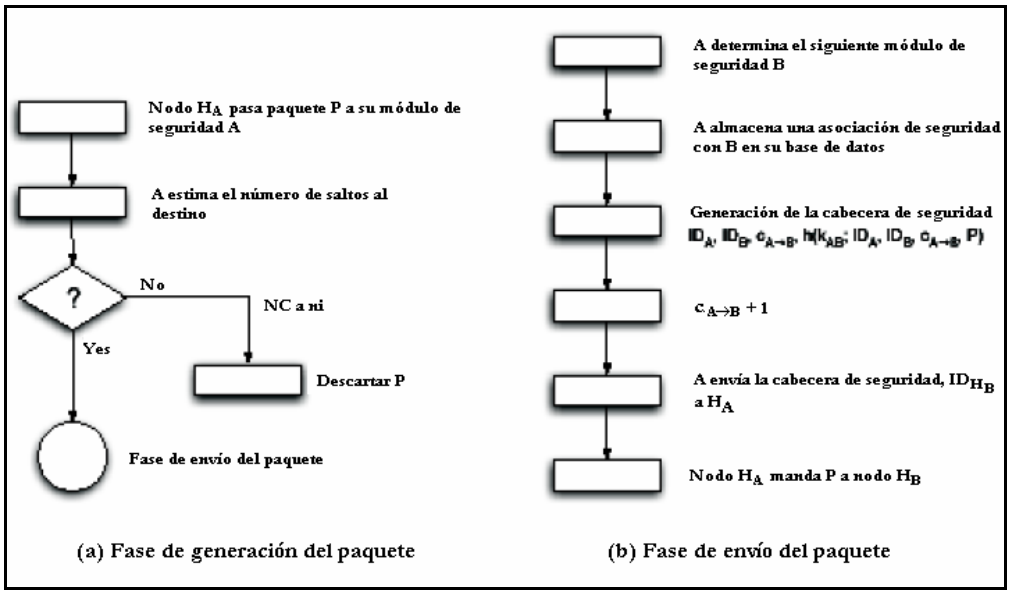
Está constituido por:

- Coprocesador de seguridad, que contiene: un procesador, memoria, batería y circuito de detección de falsificación.
- Los nuglets se representan con contadores en un módulo hardware.
- Para el cobro por envío el módulo de seguridad reclama un acuse de recibo del siguiente nodo antes de incrementar el contador.
- El siguiente nodo es animado a enviar el acuse porque recibe unos pocos nuglets.

Almacena:

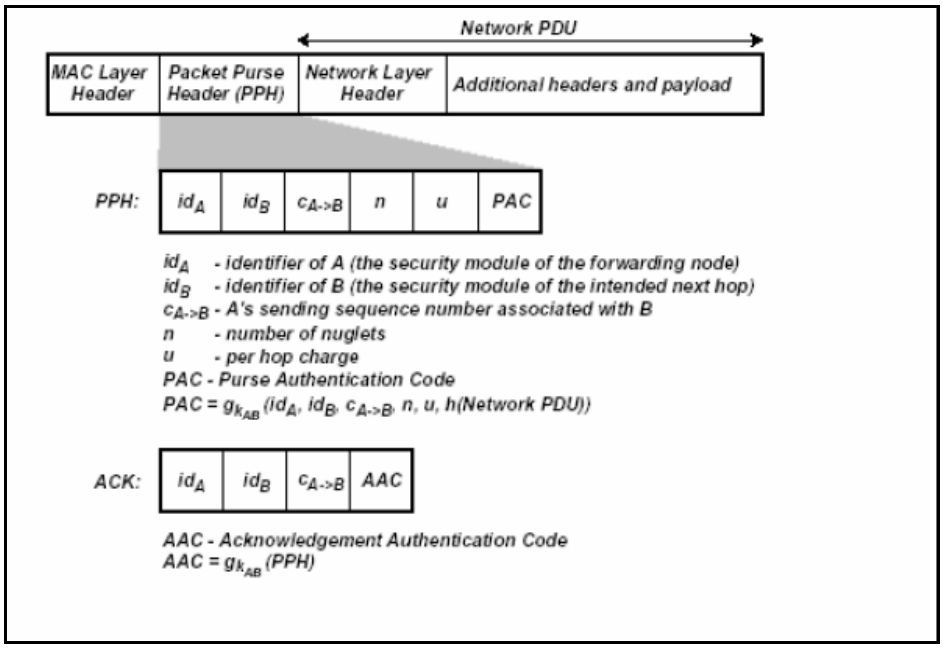
- Identificador del nodo.
- Certificado de clave pública del nodo firmado por la AC (empresa que construye los módulos).
- Las firmas de las demás ACs.
- Las claves de sesión compartidas con los vecinos.
- Números secuenciales ($c_{A>B}$, $c_{A<B}$, $c_{B>A}$, $c_{B<A}$) entre vecinos para detectar reenvíos de paquetes. Estos números se inicializan al azar y con el protocolo de hello se fija la secuencia de forma que $c_{A>B} = c_{B<A} + 1$.
- Contador de deuda $d_{A>B}$.

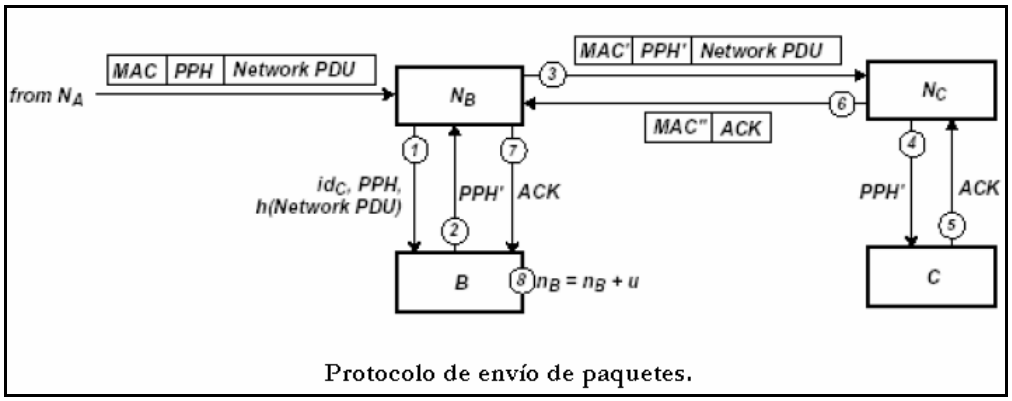




(a) Fase de generación del paquete

(b) Fase de envío del paquete



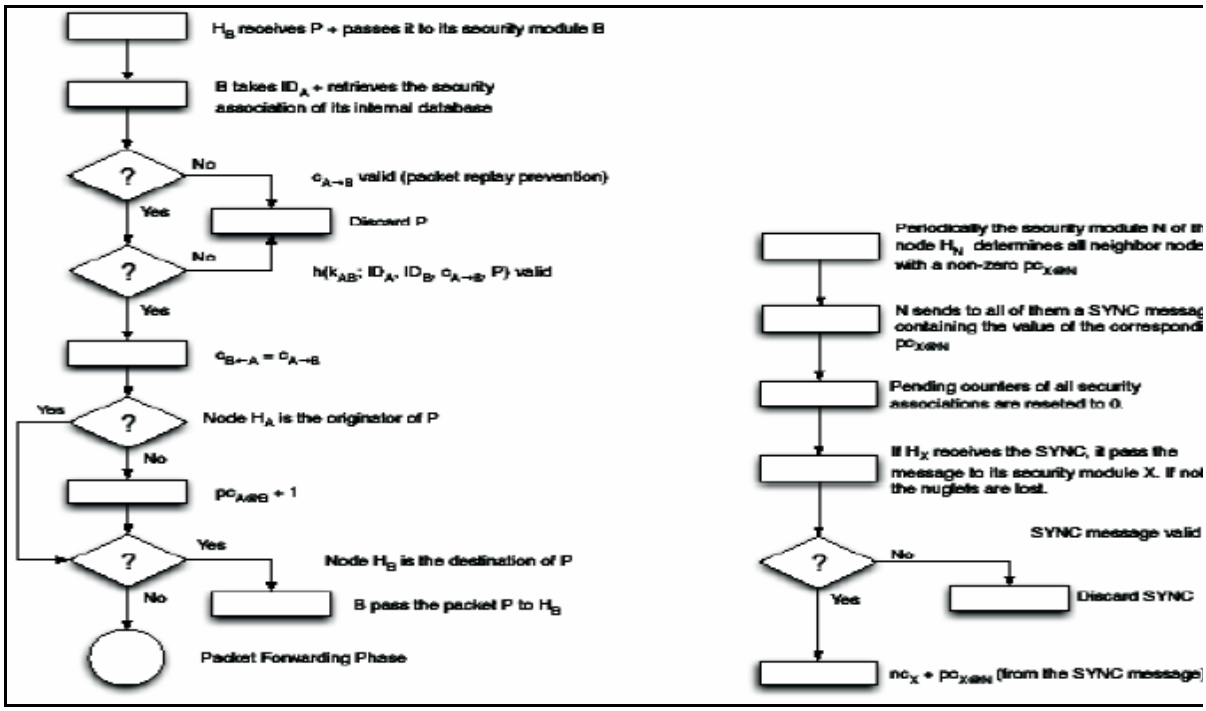


Protocolo de envío de paquetes.

2.7.5.1.1.4. Protocolo de envío de paquetes.

Vada nodo intermedio B:

- Comprueba el número secuencial.
- Actualiza $c_{A>B}$ con el número recibido.
- Comprueba la autenticidad del PPH con el código de autenticación.
- Calcula un nuevo PPH (PPH') incluyendo id_B, id_C y $C_{B>C}$, disminuyendo el número de nuglets según el cargo fijo, y calculando el código de autenticación con función hash y k_{BC}.
- Incrementa $C_{B>C}$.
- Almacena y envía a N_B PPH'.



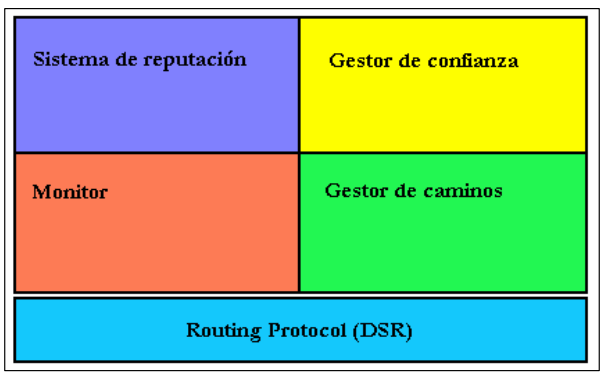
2.7.5.1.2. Confidant (Cooperation of nodes, fairness in dynamic ad-hoc networks).

Características:

- Trata de detector nodos atacantes mediante la combinación de monitorizado y el establecimiento de rutas que eviten nodos egoístas
- Causa una reacción en los demás nodos que se traduce en una desventaja para el nodo egoísta ya que los paquetes de los nodos atacantes no son enviados por los nodos honestos.
- Las relaciones de confianza y las decisiones sobre el enrutamiento se basan en las experiencias propias, observaciones de otras experiencias, e informes sobre comportamientos de otros nodos.
- Soporta hasta un 60 % de nodos egoístas,
- Si un nodo es acusado erróneamente, o se arrepiente y se comporta de forma honesta durante cierto tiempo, se lleva a cabo su re-socialización y reintegración en la red.

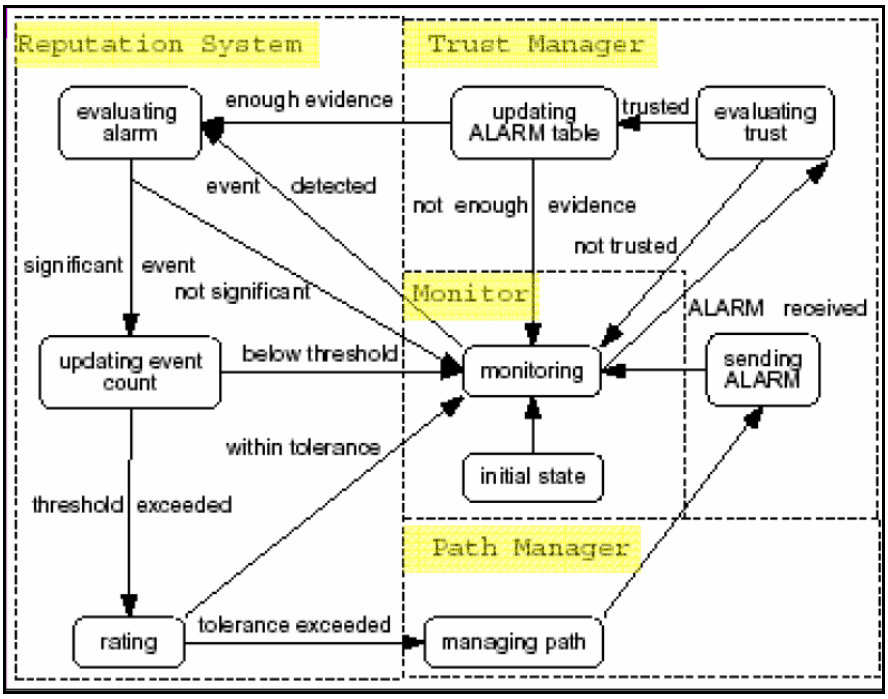
Componentes presentes en cada nodo:

- Sistema de reputación.



- Monitor.
- Gestor de confianza.
- Gestor de caminos.

Se basa en DSR.



El monitor es el vigilante del vecindario:

- Los nodos con más probabilidad de detectar un comportamiento incorrecto son los nodos del vecindario del atacante, y también la fuente y/o el destino si detectan un comportamiento inusual o no obtienen respuestas adecuadas.
- Los nodos pueden detectar desviaciones mediante experiencias propias o la escucha de la transmisión del siguiente nodo en la ruta, ya que si se mantiene una copia de un paquete mientras se escucha la transmisión del siguiente nodo, cualquier cambio de contenido puede ser detectado.
- Registra las desviaciones del comportamiento normal, y en cuanto ocurre un mal comportamiento, llama al gestor de confianza y al sistema de reputación.

El gestor de confianza:

- Trabaja con mensajes ALARMA entrantes y salientes.
- Los mensajes ALARMA son enviados por el gestor de confianza de un nodo para avisar a otros sobre la presencia de nodos atacantes, tras haber experimentado, observado o recibido un informe de comportamiento deshonesto.
- Los receptores de estos mensajes se llaman amigosm y son administrados en una lista de amigos.
- Cada nodo siempre debe comprobar el nivel de confianza de la fuente de una ALARMA antes de producir una reacción.
- Para la gestión de confianza de esas ALARMAs se usa un mecanismo similar al gestor de confianza definido en PGP.
- Se examina el nivel de confianza de todas las firmas adjuntas a un mensaje, y se calcula una puntuación de la validez, P.e., dos firmas marginalmente confiables pueden ser equivalente a una sola firma de total confianza. El esquema es ajustable de forma que puede requerir un número diferente de firmas marginalmente confiables para igualarla a una clave válida.

El gestor de confianza consiste en los siguientes componentes:

- Una tabla de alarmas con información sobre alarmas recibidas.
- Una tabla de confianza que gestiona los niveles de confianza de cada nodo para determinar la confianza de una alarma.
- Una lista de amigos con todos los amigos a los que un nodo potencialmente envía alarmas.

Una vez comprobado el nivel de confianza de una alarma, se envía al sistema de reputación.

El sistema de reputación gestiona una tabla de nodos y sus puntuaciones:

- Hace uns tasación de la calidad de los nodos.
- Para evitar una tasación centralizada, se mantienen listas de tasaciñon locales y/o listas negras en cada nodo, que se intercambian con amigos.
- En las rutas enviadas, el nodo origen puede incluir ovejas negras a evitar en el enrutamiento, que también alarma a los nodos intermedios.

- Los nodos comprueban los nodos con mala puntuación en la lista negras antes de enviar algo. El problema de distinguir entre nodos atacantes acusados y demostrados, e.d., evitar falsas acusaciones, puede disminuirse mediante listas de recuperación de nodos que se han portado bien durante un período especificado de tiempo.

- Se supone que el mal comportamiento será la excepción, por lo que el sistema de reputación se construye por experiencia negativa en lugar de por impresiones positivas.

- La puntuación de un nodo baja sólo si hay suficiente evidencia de mal comportamiento, y cuando ha ocurrido un número de veces que excede un umbral. La tasación es entonces cambiada de acuerdo con una función que asigna diferentes pesos al tipo de detección de comportamiento, p.e. mayor peso a la propia experiencia, y menor peso a las observaciones del vecindario, y menor aún a la experiencia informada.

- Si la tasación de un nodo se ha deteriorado tanto que ha caído por debajo de un rango tolerable, se activa el gestor de caminos.

El gestor de caminos tiene las siguientes funciones:

- Ordenación de caminos según la métrica escogida p.e., reputación de los nodos del camino.

- Borrado de caminos con nodos atacantes.

- Acciόm tras la recepción de una petición de ruta de un nodo atacante (p.e. ignorar, no enviar ninguna respuesta).

- Acción tras la recepción de petición de ruta hacia un nodo atacante (p.e. ignorar, alertar a la fuente).

2.7.5.1.3. Esquemas basados en tokens.

- Yang, Meng y Lu sugieren un mecanismo donde cada nodo debe tener un vale o token válido para participar en la red.

- Dichos vales son validados por los vecinos de cada nodo basándose en su participación activa en la red.

- Los vales han de ser renovados de manera que el período de validez depende del comportamiento correcto del nodo, y un nodo honesto puede acumular crédito y renovar con menos frecuencia su vale.

- El mecanismo incluye 4 componentes:

- Verificación del vecino, a través de la cual cada nodo comprueba si sus nodos vecinos son legítimos.

- Monitorización del vecino, que permite a cada nodo controlar el comportamiento de cada nodo en la red y detectar ataques de nodos atacantes.

- Reacción a la intrusión, que asegura a la generación de alertas y el aislamiento de atacantes.

- Reacción a la intrusión, que asegura la generación de alertas y el aislamiento de atacantes.

- Enrutamiento básico mejorado, incluyendo extensiones de seguridad.

Un token válido se construye usando una firma de grupo mediante un mecanismo basado en la compartición de secretos polinomial, que asegura que al menos k vecinos acuerdan distribuir o renovar el token. La complejidad de la actualización de la clave mediante un esquema umbral de Shamir, y el requerimiento de al menos k nodos que firmen cada token son incompatibles con la alta movilidad de cualquier Manet grande y densa. Por tanto, este esquema es más adecuado para Manets con poca movilidad.

2.7.6. Criptografía.

La criptografía es una ciencia cuyo objeto es transformar mediante convenciones secretas denominadas claves, informaciones o señales claras en informaciones o señales ininteligibles por terceros que no conozcan el secreto, o realizar la operación inversa gracias a medios, hardwares o softwares diseñados con este fin. La criptografía permite detectar la pérdida de integridad de informaciones, de autentificar interlocutores y proteger la confidencialidad de las informaciones.

¿En qué se basa la seguridad de la criptografía?

La seguridad de la criptografía se basa en tres factores:

- La calidad de los algoritmos utilizados. No es indispensable que estos algoritmos sean confidenciales, pero deben apoyarse en preguntas consideradas difíciles de resolver por los matemáticos, si no conocen un secreto. Este secreto se denomina clave.

- La implementación de estos algoritmos. Es mucho más fácil evitar una mala implementación de un algoritmo que “romper” el propio algoritmo.

- La gestión de las claves. Cuando se utiliza un mismo secreto entre dos usuarios (sistema simétrico), la multiplicación del número de usuarios aumenta el número de secretos que compartir de manera cuadrática. En

cambio, la utilización de un sistema donde no se necesita compartir un secreto (sistema asimétrico) facilita la gestión de los secretos pero no evita la protección de la integridad de las claves ni la exigencia de protección de la confidencialidad de los secretos propios de los usuarios.

¿Qué es una clave pública?

Se distinguen dos tipos de criptografía: la criptografía simétrica, denominada de clave secreta, y la criptografía asimétrica, denominada de clave pública.

El principio de la criptografía de clave secreta consiste en utilizar un solo secreto o una misma clave para cifrar y descifrar las informaciones. La criptografía de clave pública utiliza una clave diferente en emisión y en recepción. Sin embargo, el par de claves utilizado es criptográficamente indisociable: consta de una parte privada que es secreta y de una parte pública, cuya confidencialidad no es necesaria.

¿Cómo la criptografía de clave pública permite firmar digitalmente?

La criptografía de clave pública se caracteriza por la utilización de dos claves matemáticamente asociadas: la clave privada y la clave pública. La clave privada permite generar la firma. Es propia al firmante y debe mantenerse secreta. La clave pública se utiliza para verificar una firma. Puede ser publicada pero debe obligatoriamente permanecer entera.

El propietario de un par de claves (un par compuesto de una clave pública y de una clave privada) es el único que puede utilizar la clave privada y ejecutar la operación de generación de firma digital, mientras que la clave pública puede ser utilizada por varios destinatarios que pueden así verificar la firma.

¿Cómo la criptografía de clave pública permite cifrar informaciones?

Cuando varios usuarios utilizan la criptografía para efectuar intercambios seguros, comparten una clave secreta. Un mismo usuario puede comunicar con grupos diferentes que, al no tener acceso a los mismos tipos de información, no deben compartir el mismo secreto. Esto lleva a utilizar un gran número de claves, ya que este esquema se expande a una gran comunidad de usuarios. La gestión, la garantía de confidencialidad y de integridad de las claves secretas crea entonces dificultades en términos de recursos y de organización.

La criptografía de clave pública permite resolver estos problemas, en la medida en que cada usuario dispone de una clave privada y de solamente una. Así, cuando un usuario desea establecer una convención secreta, utiliza la clave pública de su interlocutor y pone en práctica un mecanismo criptográfico asimétrico.

¿Qué es una función hash?

Una función hash permite crear un «condensado» del mensaje que debe firmarse denominado resumen. Este tipo de función criptográfica está diseñado de manera que una modificación, incluso ínfima, del mensaje inicial conlleve una modificación del resumen.

Una función hash es tal, que resulta imposible encontrar un mensaje inicial a partir de su resumen o fabricar dos mensajes ininteligibles que tengan el mismo resumen.

¿Qué significa "firmar digitalmente un documento"?

La generación de la firma numérica de un documento consiste en calcular el resumen de un documento y en cifrar el resumen utilizando una clave privada. El resultado obtenido se denomina firma digital y puede adjuntarse al documento que debe firmarse.

Un usuario firma un documento digitalmente para que sus destinatarios puedan detectar una pérdida de integridad y autentificar el origen.

¿Qué significa "verificar la firma de un documento"?

La verificación de la firma de un documento consiste en "descifrar" la firma gracias a la clave pública del remitente, y en comparar el claro obtenido con el resumen calculado a partir del mensaje efectivamente recibido. Si los dos resúmenes son idénticos, entonces la firma es válida.

La verificación de una firma garantiza la detección de la pérdida de integridad del mensaje y la autenticación del origen del mensaje. La verificación garantiza la autenticación sólo si el autor es realmente el único poseedor del secreto que es la clave privada de firma.

¿Por qué es necesario diferenciar el par de claves por tipo de aplicación?

Se distinguen por lo menos tres tipos de pares de claves que responden a necesidades de seguridad diferentes:

- Los pares de claves de firma, que permiten garantizar la integridad y autentificar el origen de un mensaje,
- Los pares de claves de intercambio de clave, destinados a proteger un intercambio de clave secreta (utilizada para cifrar un mensaje),
- Los pares de claves de cifrado, utilizados para cifrar directamente mensajes (sobre todo de tamaño pequeño).

Cuando un par de claves se utiliza para cifrar o intercambiar claves secretas, las informaciones protegidas son ininteligibles sin el conocimiento del secreto que es la clave privada de cifrado. Pero este par de claves puede estar comprometidas, indisponibles, tener los derechos vencidos, etc. En tales casos,

se debe poner en práctica medios de recuperación del par de claves a fin de garantizar la recuperación de las informaciones protegidas.

La protección de una información por una firma digital no hace que la información sea ininteligible. Por lo tanto, en caso de indisponibilidad de la clave privada de firma, no se necesita en absoluto proceder a la recuperación del par de claves para recuperar la información. Al contrario, nadie debe ser capaz de reproducir el par de claves de firma, y por lo tanto, de generar una firma a nombre de otra persona.

Para satisfacer estas exigencias de seguridad diferentes, los pares de claves de firma, de intercambio de claves y de cifrado deben ser diferentes.

¿Qué es un certificado de clave pública?

Un certificado de clave pública crea un fuerte lazo entre una clave pública, la identidad de su propietario y el uso posible de la clave (firma, intercambio de claves, cifrado, etc.).

Este lazo es establecido por una autoridad de confianza denominada autoridad de certificación que garantiza la veracidad de las informaciones contenidas en el certificado.

Para ello, la autoridad de certificación firma con su propia clave privada un mensaje que contiene particularmente la identidad del propietario de la clave, la clave pública, el uso posible de la clave, según una política de certificación definida. Este mensaje particular se denomina certificado de clave pública y está normalizado por el formato X.509v3.

La seguridad de aplicaciones tales como el comercio electrónico, las mensajerías electrónicas, el acceso a servidores de información (Web), la gestión de infraestructuras de red está garantizada por la utilización de certificados.

¿Qué es una autoridad de registro?

Una autoridad de registro representa el punto de contacto entre el usuario y la autoridad de certificación. Cuando un usuario desea obtener un certificado, debe presentarse y efectuar la solicitud ante una autoridad de registro.

El registro de un usuario tiene lugar solamente tras la verificación de informaciones propias al solicitante de certificado. El tipo de las informaciones verificadas depende de la política de certificación puesta en práctica: Estas informaciones pueden consistir en la prueba de la identidad del solicitante, la prueba de posesión de la clave privada correspondiente a la clave pública que ha de certificarse, una autorización de solicitud de certificado, una prueba de la función ocupada por el solicitante, etc.

Una vez verificadas estas informaciones, se efectúa el registro y se comunican a la autoridad de certificación solamente las informaciones necesarias para la certificación.

¿Qué es una autoridad de certificación?

Una autoridad de certificación genera certificados por cuenta de usuarios.

Al recibir el certificado emitido por una autoridad de registro, la autoridad de certificación firma el certificado con su propia clave privada, garantizando así la integridad del certificado, y la veracidad de las informaciones contenidas.

La integridad de la clave pública de la autoridad de certificación, y la confidencialidad y la integridad de su clave privada son condiciones indispensables para la seguridad del sistema.

¿Qué es un servicio de publicación de certificados?

Un servicio de publicación pone a disposición de una comunidad de usuarios los certificados de claves públicas emitidos por una o varias autoridades de certificación. También publica una lista de certificados revocados. El medio utilizado para publicar los certificados pueden ser uno o varios anuarios, un documento en papel, un servidor de información (Web), un disquete, etc. Bajo ciertas condiciones, el servicio de publicación debe estar al día, accesible y disponible.

¿Qué es una revocación?

Cuando un usuario pierde o divulga su clave privada, o las informaciones contenidas en un certificado son falsas (falsificación de la identidad, pérdida de la integridad de la clave pública contenida en el certificado), el certificado ya no es de confianza y su utilización ya no puede garantizar ninguna función de seguridad.

En tales casos, la autoridad de certificación revoca el certificado comprometido y transmite la información al servicio de publicación. El servicio de publicación publica entonces dicho certificado en la lista de los certificados revocados.

El usuario de un certificado tiene el deber de verificar que el certificado que piensa utilizar no ha sido revocado.

¿Qué es una infraestructura de gestión de claves (IGC)?

Una infraestructura de gestión de claves es un conjunto de componentes de tecnologías de la información. Este conjunto contribuye a la seguridad de los pares de claves generando y asegurando la gestión completa de certificados de

claves públicas.

Consta de autoridades de certificación, de autoridades de registro y de un servicio de publicación.

¿Qué servicios ofrece una IGC?

Una infraestructura de gestión de claves ofrece un conjunto de servicios por cuenta de sus usuarios.

Los servicios son principalmente los siguientes:

- Registro de los usuarios.
- Generación de certificados.
- Revocación de certificados.
- Publicación de los certificados válidos y revocados.
- Identificación y autenticación de los usuarios.
- Archivo de los certificados.

La IGC realiza también un conjunto de operaciones para satisfacer necesidades internas de la infraestructura (imputabilidad de las operaciones, identificación y autenticación de los profesionales, etc.) y puede eventualmente ofrecer servicios complementarios a los usuarios (tales como la generación de pares de claves de autenticación, la recuperación de claves por cuenta de usuarios, la emisión de un fechado de confianza, etc.).

En el documento, se presenta una clasificación por tipo de servicios suministrados por una IGC (servicios vitales de seguridad, servicios elementales y complementarios).

¿Cuáles son los indicadores de confianza de una IGC?

La confianza otorgada a un certificado depende directamente de la confianza otorgada a la infraestructura entera. Sin embargo, la naturaleza distribuida del sistema de información que constituye la IGC dificulta la evaluación del nivel de confianza otorgado a una infraestructura de gestión de claves.

Existen varios indicadores para ayudar al usuario en esta evaluación:

La política de certificación puesta en práctica por la infraestructura,

- La declaración relativa a los procedimientos de certificación que aplica,
- El hecho de que la IGC esté en conformidad con los documentos que formalizan un nivel de seguridad según criterios de evaluación reconocidos (perfiles de protección),
- Eventualmente, el régimen legal de la infraestructura, si existen esquemas de acreditación de las IGC (como por ejemplo la acreditación de las autoridades de certificación del sector bancario).

¿Qué es una "política de certificación"?

Una política de certificación describe el conjunto de las reglas que definen el tipo de aplicación a las cuales está adaptado un certificado. Un certificado de clave pública contiene la identificación de la política de certificación con la cual fue emitido, y según la cual debe utilizarse.

El grupo PKIX de la IETF (Internet Engineering Task Force), definió un formalismo de descripción de las políticas de certificación. Este formalismo, en curso de normalización en la IETF, da el conjunto de las especificaciones que deben suministrarse para describir una política de certificación en términos de responsabilidades (legales, jurídicas y financieras), de funcionalidades y de administración.

¿Qué es una "declaración relativa a los procedimientos de certificación"?

Una declaración relativa a los procedimientos de certificación enuncia las prácticas utilizadas por la IGC en la gestión de los certificados, prácticas que dependen de la política de certificación puesta en práctica. Una IGC debe publicar esta declaración a fin de describir las modalidades de funcionamiento de los servicios que brinda.

Una declaración relativa a los procedimientos de certificación puede contener, por ejemplo, el tipo de los documentos que deben suministrarse para el registro, el nombre de las personas a contactar para una revocación, las horas de apertura de las diferentes autoridades.

¿Qué es el documento "Procedimientos y Políticas de Certificación de Claves" (PC2)?

Este documento describe, en un formalismo normalizado en la IETF ("Internet Engineering Task Force") las políticas de certificación adaptadas a dos tipos de necesidades (necesidad de autenticación o de confidencialidad) y a distintos niveles de sensibilidad en cuanto a las informaciones procesadas.

El documento PC2 propone también una guía para la redacción de una declaración relativa a los procedimientos de certificación.

Este documento es el resultado de trabajos llevados a cabo en el grupo Ad Hoc de Mensajería Segura de la Subcomisión de Criptografía de la CISSI (Comisión Interministerial de la Seguridad de los Sistemas de Información), para las necesidades de las administraciones francesas. Puede utilizarse para la descripción de las políticas de certificación de infraestructuras de distinta naturaleza.

¿Qué es una certificación cruzada?

Cuando los usuarios de diferentes IGC intercambian informaciones, deben utilizar certificados emitidos por autoridades de certificación en las cuales en principio no tienen por qué tener confianza. Por definición, un usuario

tiene confianza en su propia IGC. Si su propia IGC «declara» que confía en la otra IGC, entonces el usuario también podrá confiar en ella.

Para ello, las infraestructuras pueden convenir entre ellas en acuerdos de reconocimiento, según las políticas de certificación puestas en práctica.

Cuando una autoridad otorga su confianza a un tipo de certificado emitido por otra autoridad de certificación (por ejemplo porque las políticas de certificaciones son idénticas o equivalentes), certifica la clave pública de esta autoridad. De la misma forma, la segunda puede reconocer certificados de la primera autoridad, constituyendo así un reconocimiento mutuo para un tipo de certificado.

¿Qué es el Perfil de Protección Infraestructuras de Gestión de Claves?

Según los Criterios Comunes de Evaluación de la seguridad de las Tecnologías de la Información, en curso de normalización en la ISO, un perfil de protección pone en evidencia, para una categoría de objetivos de evaluación, las exigencias funcionales y de garantía que permiten responder a las amenazas identificadas, a fin de alcanzar un conjunto de objetivos de seguridad.

El perfil de protección de una infraestructura de gestión de claves define como objetivo de evaluación el conjunto de los componentes de una infraestructura de gestión de claves.

A partir de este perfil, una IGC particular define el objetivo de seguridad correspondiente a su propio sistema. Su conformidad con las exigencias impuestas por el perfil le permite aportar la prueba que está en conformidad con el nivel de garantía definido en el PP_IGC.

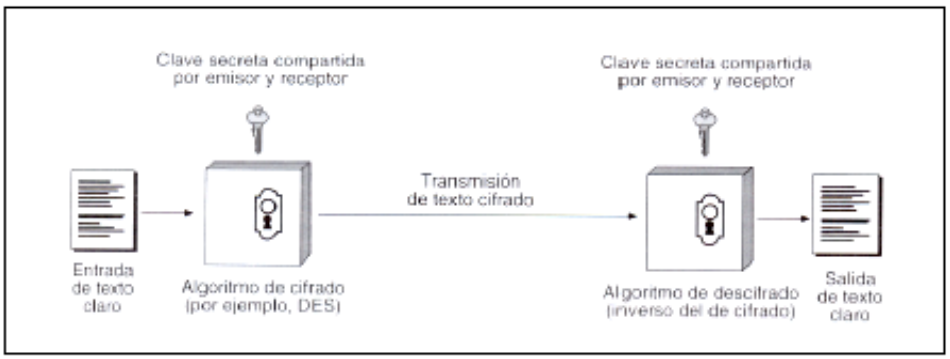
Este perfil de protección es el resultado de trabajos llevados a cabo en el grupo Ad Hoc de Mensajería Segura. Corresponde a las exigencias requeridas para procesar informaciones en la administración francesa desde un nivel de sensibilidad poco elevado hasta un nivel clasificado como de defensa.

2.7.6.1. Criptografía simétrica.

Utiliza una misma clave para cifrar y para descifrar mensajes. Las dos partes que se comunican se ponen de acuerdo de antemano sobre la clave a usar. Una vez que ambas tienen acceso a esta clave, el remitente cifra un mensaje usándola, lo envía al destinatario, y éste lo descifra con la misma. Un buen sistema de cifrado pone toda la seguridad en la clave y ninguna en el algoritmo. En otras palabras, no debería ser de ninguna ayuda para un hacker o cracker conocer el algoritmo que se está usando. Sólo si el atacante obtuviera la clave, le serviría conocer el algoritmo. Los algoritmos de cifrado usados por ejemplo en el sistema GNU, GnuPG tienen estas propiedades. Dado que toda la seguridad está en la clave, es importante que sea muy difícil descifrar el tipo de clave. Esto quiere decir que el abanico de claves posibles,

es decir, el espacio de posibilidades de claves, debe ser amplio. Actualmente los computadores y servidores pueden adivinar claves con extrema rapidez, y ésta es la razón por la cual el tamaño de la clave es importante en los criptosistemas modernos. El algoritmo de cifrado DES usa una clave de 56 bits, lo que significa que hay 2 elevado a 56 claves posibles. 2 elevado a 56 son 72.057.594.037.927.936 claves. Esto representa un número muy alto de claves, pero un PC de uso general puede comprobar todo el espacio posible de claves en cuestión de días. Una máquina especializada lo puede hacer en horas. Por otra parte, algoritmos de cifrado de diseño como 3DES, Blowfish e IDEA usan todos claves de 128 bits, lo que significa que existen 2 elevado a 128 claves posibles.

Esto representa muchas más claves, y aun en el caso de que todos los PCs del planeta estuvieran cooperando, todavía tardarían más tiempo que la misma edad del universo en encontrar la clave. Incluso en la actualidad se pueden encontrar en el mercado claves a 256 bits, 512 bits y más. El principal problema con los sistemas de cifrado simétrico no está ligado a su seguridad, sino al intercambio de claves. Una vez que el remitente y el destinatario hayan intercambiado las claves pueden usarlas para comunicarse con seguridad, pero ¿qué canal de comunicación que sea seguro han usado para transmitirse la clave entre sí? Sería mucho más fácil para un atacante intentar interceptar una clave que probar las posibles combinaciones del espacio de claves. Es aquí donde entran la criptografía asimétrica y la criptografía híbrida.



2.7.6.1.1. Algoritmos de cifrado simétrico

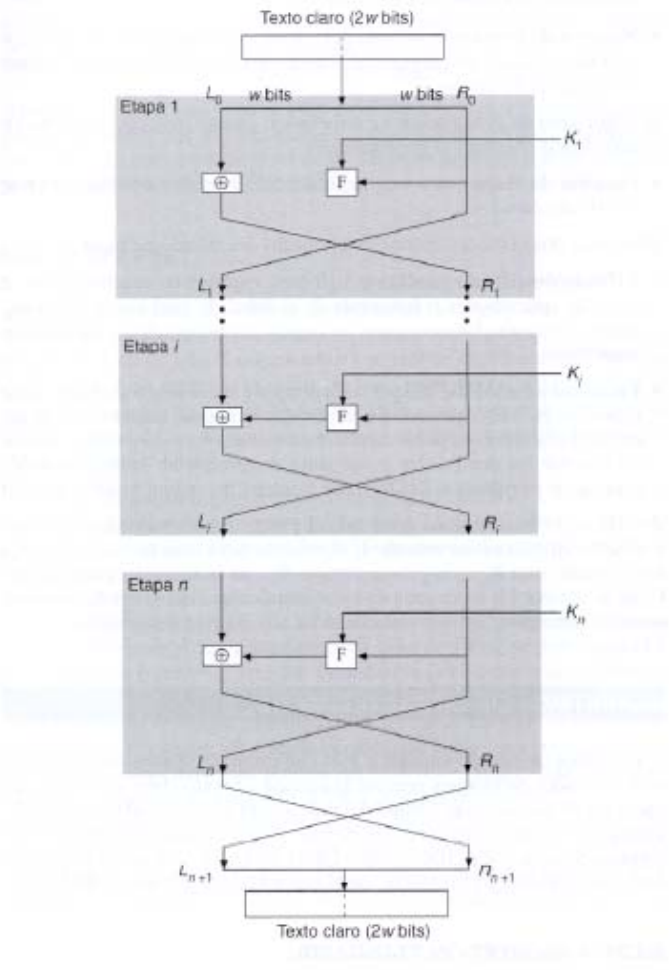
Los algoritmos de cifrado simétrico más comúnmente usados son los cifradores de bloques. Un cifrador de bloques procesa la entrada de texto claro en bloques de tamaño fijo y genera un bloque cifrado del mismo tamaño para cada texto claro.

2.7.6.1.2. DES (Data Encryption Standard)

El esquema de cifrado más extendido se basa en el DES (Data Encryption Standard) adoptado en 1977 por el Nacional Bureau of Standards, ahora el NIST (Nacional Institute of Standards and Technology), como Federal Information Processing Standard 46.

2.7.6.1.3. Descripción del algoritmo.

El texto claro tiene una longitud de 64 bits y la clave, de 56; si el texto en claro es más largo se procesa en bloques de 64 bits. La estructura del DES consiste en una pequeña variación de la red de Feistel, que se muestra en la figura siguiente. Hay 16 etapas de proceso. Se generan 16 subclaves partiendo de la clave original de 56 bits, una para cada etapa.



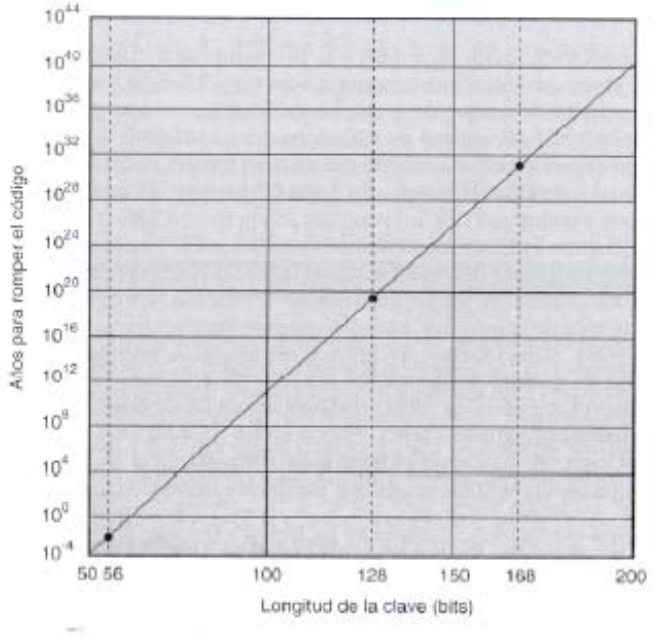
El proceso de descifrado con el DES es básicamente el mismo que el de cifrado. La regla es la siguiente: usar el texto cifrado como entrada al algoritmo del DES, pero las subclaves Ki se pasan en orden inverso. Es decir, en la primera etapa se usa K16, K15 en la segunda y así sucesivamente hasta K1 en la 16ª y última.

2.7.6.1.4. Robustez del DES.

Los aspectos de robustez del DES se engloban en dos categorías: aspectos sobre el algoritmo mismo y aspectos sobre el uso de una clave de 56 bits. Los primeros se refieren a la posibilidad de que el criptoanálisis se realice explotando las características del algoritmo DES. A lo largo de los años, se han

intentado encontrar debilidades que explotar en el algoritmo, lo que ha hecho del DES el algoritmo de cifrado existente más estudiado. A pesar de los numerosos enfoques, nadie ha conseguido descubrir ninguna debilidad grave en el DES. Un aspecto de mayor importancia es la longitud de la clave. Con una clave de 56 bits, hay 256 claves posibles, que es aproximadamente 7,2 X 1016 claves. Por este motivo, no parece práctico un ataque de fuerza bruta. Suponiendo que, en promedio, se tiene que intentar la mitad del espacio de claves, una única máquina que realice un cifrado DES por microsegundo tardaría más de mil años en romper el cifrado.

En cualquier caso, la suposición de un cifrado por microsegundo es demasiado conservadora. Finalmente y definitivamente, en julio de 1998, se probó que el DES no era seguro, cuando la Electronic Frontier Foundation (EFF) anunció que había roto un cifrado DES utilizando una máquina especializada <<DES craker>>, construida por menos de 250.000 dólares. El ataque duró menos de tres días. La EFF ha publicado la descripción detallada de la máquina, haciendo posible que cualquiera construya su propia craker. Naturalmente, los precios del hardware continuarán bajando mientras la velocidad irá aumentando, haciendo al DES prácticamente inútil. Es importante tener en cuenta que para que un ataque de búsqueda de clave no basta con probar todas las posibles claves. A menos que se suministre el texto claro, el analista debe ser capaz de reconocer el texto claro como tal. Si el mensaje es texto claro en inglés, entonces el resultado se obtiene fácilmente, aunque la tarea de reconocimiento del inglés tendría que estar automatizada. Si el mensaje de texto se ha comprimido antes del cifrado, entonces el reconocimiento es más difícil. Y si el mensaje es de un tipo más general de datos, como un fichero numérico, y ha sido comprimido, el problema es aún más difícil de automatizar. Pero eso, para complementar el enfoque de fuerza bruta, se necesita algún grado de conocimiento sobre el texto claro esperado y alguna forma de distinguir automáticamente el texto claro de lo que no lo es. El enfoque de la EFF trata también este tema, e introduce algunas técnicas automatizadas que serían efectivas en muchos contextos. Como punto final, si la única forma de ataque a un algoritmo de cifrado es la fuerza bruta, entonces la manera de contrarrestar este ataque es obvia: usar claves más largas. Para tener una idea del tamaño de clave necesario, usaremos el craker de la EFF como base de nuestras estimaciones. El craker de la EFF era un prototipo, y se puede suponer que con la tecnología actual es rentable construir una máquina más rápida. Si asumimos que un dispositivo como el craker puede realizar un millón de descifrados por us, entonces se tardaría alrededor de diez horas en descifrar un código DES. Esto constituye un incremento de velocidad aproximadamente un factor de siete comparado con los resultados de la EFF. Usando este ratio, la figura 3 muestra cuanto tardaría en romper un algoritmo del estilo del DES en función del tamaño de la clave. Por ejemplo, para una clave de 128 bits, que es común entre los algoritmos actuales, se tardaría 1018 años en romper el código usando el craker de la EFF. Incluso, aunque se aumentara la velocidad del craker en un factor de un trillón (1012), todavía se tardaría un millón de años en romper el código. Así que una clave de 128 bits garantiza que el algoritmo es inexpugnable por la fuerza bruta.



2.7.6.1.5. Triple DES

El triple DES (3DES) se estandarizó inicialmente para aplicaciones financieras en el estándar ANSI X9.17 en 1985. el 3DES se incorporó como parte del DES en 1999, con la publicación de FIPS PUB 463. El 3DES usa tres claves y tres ejecuciones del algoritmo DES. La función sigue la secuencia cifrardescifrarcifrar (EDE: encryptdecryptencrypt):

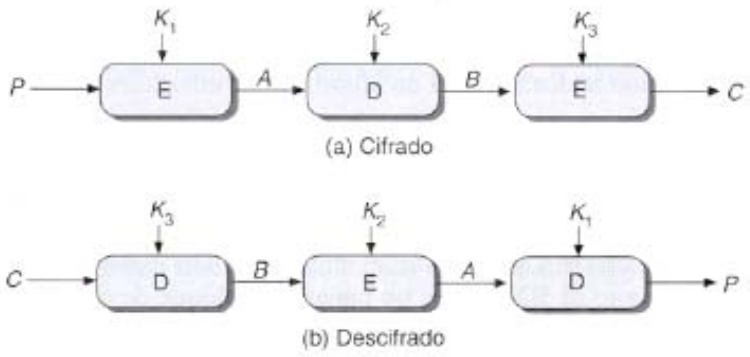
$$C = E_{K3}[D_{K2}[E_{K1}[P]]]$$

Donde

C = texto cifrado
P = texto claro
$E_K[X]$ = cifrado de X usando la clave K
$D_K[Y]$ = descifrado de Y usando la clave K

El descifrado es simplemente la misma operación con las claves en orden inverso:

$$P = D_{K1}[E_{K2}[D_{K3}[C]]]$$



2.7.6.1.6. Triple DES

El descifrado del segundo paso no es significativo en términos criptográficos. Su única ventaja es que permite a los usuarios del 3DES descifrar datos cifrados por usuarios del DES:

C = EK1 [DK1 [EK1 [P]]] = EK1 [P]

Con tres claves diferentes, el 3DES tiene una longitude efectiva de clave de 168 bits. El FIPS 463 también permite el uso de dos claves, con K1 = K3, lo que proporciona una longitud de clave de 112 bits. El FIPS 463 incluye las siguientes directrices para el 3DES:

- El 3DES es el algoritmo de cifrado simétrico oficial del FIPS.
- El DES original, que usa una única clave de 56 bits, se mantiene sólo para los sistemas existentes. Las nuevas adquisiciones deberían admitir 3DES.
- Se apremia a las organizaciones gubernamentales con sistemas que usan DES a migrar a 3DES.

Se prevé que el 3DES y el AES (Advanced Encryption Standard) coexistirán como algoritmos oficiales del FIPS, permitiendo una transición gradual hacia el AES.

Es fácil observar que el 3DES es un algoritmo robusto. Debido a que el algoritmo criptográfico que lo sustenta es el DES, el 3DES resulta igual de resistente al criptoanálisis basado en el algoritmo que el DES. Es más, con una clave de 168 bits de longitud, los ataques de fuerza bruta son efectivamente imposibles.

2.7.6.1.7. AES (Advanced Encryption Standard)

El 3DES tiene dos atractivos que aseguran su uso durante los próximos años. Primero, con su longitud de clave de 168 bits evita la vulnerabilidad al ataque de fuerza bruta del DES. Segundo, el algoritmo de cifrado que usa es el mismo que en el DES. Este algoritmo ha estado sujeto a más escrutinios que ningún otro durante un largo periodo de tiempo, y no se ha encontrado ningún ataque criptoanalítico efectivo basado en el algoritmo que no sea la fuerza bruta. Por lo tanto, hay un alto grado de seguridad en la resistencia al criptoanálisis del 3DES. Si la seguridad fuera la única consideración, el 3DES sería una elección adecuada para un algoritmo de cifrado estándar durante las primeras décadas. El inconveniente principal del 3DES es que el algoritmo es relativamente lento en su implementación software. El DES original se diseñó para implementaciones hardware de mediados de los 70 y no produce código software eficiente. El 3DES tiene tres veces más etapas que el DES y, por ello es más lento.

Debido a este inconveniente, el 3DES no es un candidato razonable para usarlo durante mucho tiempo. Para reemplazarlo, el NIST realizó en 1997 un concurso de propuestas para el desarrollo de un nuevo estándar de cifrado avanzado (AES), que debería ser tan robusto o más que el 3DES y que mejoraría significativamente la eficiencia. Además de esos requisitos generales, el NIST especificó que el AES debía ser un cifrador simétrico de bloque con una longitud de bloque de 128 bits y permitir longitudes de clave de 128, 192 y 256 bits.

2.7.6.1.8. Descripción del algoritmo.

El AES usa una longitud de bloque de 128 bits y la longitud de la clave puede ser de 128, 192 o 256 bits. En la descripción de esta sección se asume una longitud de clave de 128 bits, que posiblemente es la más implementada. La figura siguiente muestra la estructura genera del AES. La entrada a los algoritmos de cifrado y descifrado es un solo bloque de 128 bits. En el FIPS PUB 197, este bloque se representa como una matriz cuadrada de bytes. Este bloque se copia en el vector Estado, que se modifica en cada etapa del cifrado o descifrado. Después de la última etapa, Estado se copia en una matriz de salida. De igual manera, la clave de 128 bits se representa como una matriz cuadrada de bytes. Esta clave luego se expande en un vector de palabras para la generación de claves; cada palabra tiene cuatro bytes, y el número total de palabras para generar claves es de 44 para la clave de 128 bits. El orden de los bytes dentro de una matriz se establece por columnas. Así, por ejemplo, los primeros cuatro bytes de una entrada de texto claro de 128 bits al cifrador ocupan la primera columna de la matriz in, los segundos cuatro bytes la segunda columna, y así sucesivamente. De igual forma, los primeros cuatro bytes de la clave expandida, que forman una palabra, ocupan la primera columna de la matriz w.

Los siguientes comentarios revelan algunos aspectos del AES:

- Una característica notable de su estructura es que no es una estructura Feistel. En la estructura clásica de Feistel, la mitad del bloque de datos se usaba para modificar la otra mitad, y entonces se intercambiaban

entre sí. El AES procesa todo el bloque de datos en paralelo durante cada etapa, realizando sustituciones y permutaciones.

- La clave suministrada como entrada se expande en un vector de 44 palabras de 32 bits, w[i]. Cuatro palabras diferentes (128 bits) sirven como clave de etapa en cada ronda.

- Se utilizan cuatro fases diferentes, una de permutación y tres de sustitución:

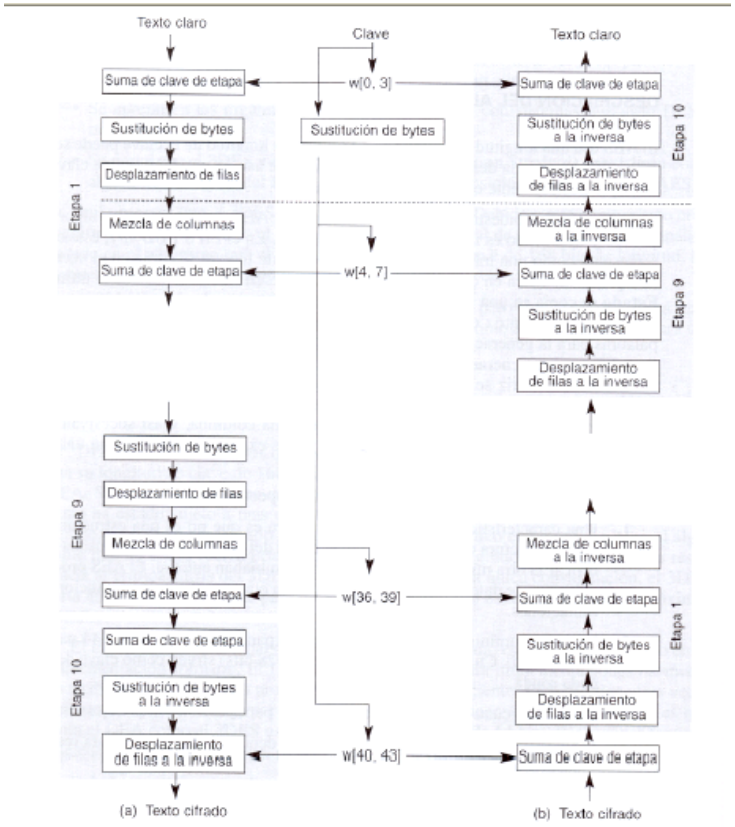
  - Sustitución de bytes: se usa una tabla, denominada caja S4, para realizar una sustitución byte a byte del bloque.

  - Desplazamiento de filas: una simple permutación realizada fila por fila.

  - Mezcla de columnas: una sustitución que altera cada byte de una columna en función de todos los bytes de la columna.

  - Suma de la clave de etapa: una simple operación XOR bit a bit del bloque actual con una porción de la clave expandida.

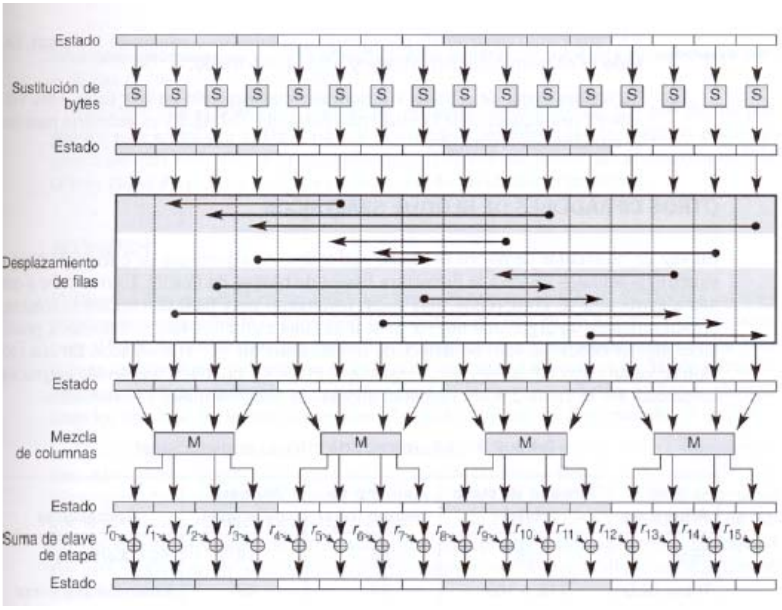
- La estructura es muy simple. Tanto para el cifrado como para el descifrado, se comienza con una fase de suma de clave de etapa, seguido de nueve etapas de cuatro fases cada una, y acaba con una décima etapa de tres fases. La figura siguiente muestra la estructura de una etapa completa de cifrado.



- Solamente la fase de suma de la clave de etapa utiliza la clave. Por esta razón el cifrador comienza y termina con una suma de clave de etapa. Cualquier otra fase, aplicada al comienzo o al final, sería reversible sin conocer la clave y por tanto añadiría inseguridad.

- La fase de suma de la clave de etapa no funcionaría por sí misma. Las otras tres fases juntas desordenan los bits, pero no proporcionan seguridad por sí mismas, porque no usan la clave. Se puede ver el cifrador como una secuencia alternativa de operaciones de cifrador XOR (suma de clave de etapa) de un bloque, seguida por un desordenamiento del bloque (las otras tres fases), seguida por un cifrado XOR, y así sucesivamente. Este esquema es eficiente y muy seguro.

- Cada fase es fácilmente reversible. Para las fases de sustitución de byte, desplazamiento de fila y mezcla de columnas, se usa una función inversa en el algoritmo de descifrado. Para la fase de suma de clave de etapa, la inversa se consigue con un XOR entre la misma clave de etapa y el bloque, usando la propiedad de que A Θ A Θ B = B.



- Como con la mayoría de los cifradores de bloque, el algoritmo de descifrado hace uso de la clave expandida en orden inverso. De todas formas, como consecuencia de la estructura particular del AES, el algoritmo de descifrado no es idéntico al de cifrado.

- Una vez se ha establecido que las cuatro fases de cada etapa son reversibles, es fácil verificar que el descifrador recupera el texto claro. La figura 5 muestra el cifrado y el descifrado desplazándose en direcciones verticalmente opuestas. En cada punto horizontal (por ejemplo, la línea discontinua la figura), Estado es el mismo para el cifrado y para el descifrado.

- La última etapa de cifrado y descifrado consiste sólo en tres fases. Otra vez, esto es consecuencia de la estructura particular del AES y es necesario para que el cifrador sea reversible.

2.7.6.1.9. Otros cifradores de bloque simétricos

En vez de reinventar de nuevo la rueda, casi todos los algoritmos de cifrado de bloque simétricos actuales utilizan la estructura básica de bloque de Feistel. Esto se debe a que dicha estructura se comprende muy bien, resultado más fácil determinar la robustez criptográfica de un algoritmo nuevo. Si se usara una estructura totalmente nueva, podría tener alguna debilidad sutil no detectada inmediatamente por el diseñador. En la tabla 1 se comparan algunas de las principales características.

Algoritmo	Tamaño de clave (bits)	Tamaño de bloque (bits)	Número de etapas	Aplicaciones
DES	56	64	16	SET, Kerberos
Triple DES	112 o 168	64	48	Financial Key management, PGP, S/MIME
AES	128, 192 o 256	128	10, 12 o 14	Destinados a sustituir DES y 3DES
IDEA	128	64	8	PGP
Blowfish	Variable hasta 448	64	16	Varios paquetes de software
RC5	Variable hasta 2048	64	Variable hasta 255	Varios paquetes de software

2.7.6.1.10. IDEA

El IDEA usa una clave de 128 bits. Difiere notablemente del DES en la función de etapa así como en la función de generación de subclaves. Para la función de etapa el IDEA no usa cajas S, sino que cuenta con tres operaciones matemáticas diferentes: XOR, suma binaria de enteros de 16 bits, y multiplicación binaria de enteros de 16 bits. Esas funciones se combinan de tal forma que producen una transformación compleja muy difícil de analizar, y por ende muy difícil para el criptoanálisis. El algoritmo de generación de subclaves se basa solamente en el uso de desplazamientos circulares pero ejecutados de manera compleja para generar un total de seis subclaves para cada una de las ocho etapas del IDEA. Debido a que el IDEA fue uno de los primeros algoritmos de 128 bits de los propuestos para remplazar al DES, ha sido sometido a considerables exámenes y por ahora parece ser muy resistente al criptoanálisis. El IDEA se usa como una alternativa en PGP (Pretty Good Privacy) y también en una serie de productos comerciales.

2.7.6.1.11. Blowfish

El Blowfish consiguió ser rápidamente una de las alternativas más populares al DES. Se diseñó para que fuera fácil de implementar y rápido en su ejecución. También es un algoritmo muy compacto que puede ejecutarse en menos de 5K de memoria. Una característica interesante es que la longitud de la clave es variable, pudiendo alcanzar hasta los 448 bits. En la práctica se usan claves de 128 bits. El Blowfish usa 16 etapas. Utiliza cajas S y la función XOR, como el DES, pero también utiliza sumas binarias. Al contrario que el DES, que utiliza cajas S estáticas, Blowfish usa cajas S dinámicas generadas como una función de la clave. Las subclaves y las cajas S se generan por la aplicación repetida del propio algoritmo a la clave. Se necesita un total de 521 ejecuciones del algoritmo de cifrado Blowfish para producir las subclaves y las cajas S. Por este motivo no es adecuado para aplicaciones en las que la clave secreta cambia frecuentemente. Este es uno de los algoritmos de cifrado simétrico más robusto hasta la fecha, porque tanto las subclaves como las cajas S se generan por un proceso de aplicaciones repetidas del propio algoritmo, lo cual modifica totalmente los bits haciendo muy difícil el criptoanálisis. Hasta ahora, se han publicado algunos artículos sobre Blowfish, sin que se hayan encontrado debilidades.

2.7.6.1.12. RC5

El RC5 se diseñó para tener las siguientes características:

- *Adecuado para hardware y software*: sólo usa operaciones computacionales primitivas que se encuentran comúnmente en los microprocesadores.
- *Rápido*: para conseguir esto, el RC5 es un algoritmo simple y orientado a palabras. Las operaciones básicas procesan palabras enteras de datos cada vez.
- *Adaptable a procesadores con diferentes tamaños de palabra*: el número de bits en una palabra es un parámetro del RC5; diferentes longitudes de palabra producen algoritmos diferentes.
- *Número variable de etapas*: el número de etapas es un segundo parámetro. Esto permite alcanzar un compromiso entre mayor rapidez y mayor seguridad.
- *Longitud de clave variable*: la longitud de la clave es un tercer parámetro. Otra vez, posibilita un acuerdo entre velocidad y seguridad.
- *Simple*: la estructura simple del RC5 es fácil de implementar y facilita la tarea de determinar la robustez del algoritmo.
- *Bajo consumo de memoria*: la poca necesidad de memoria hace que el RC5 sea adecuado para tarjetas inteligentes y otros dispositivos con restricciones de memoria.
- *Alta seguridad*: proporciona alta seguridad con los parámetros adecuados.
- *Rotaciones dependientes de los datos*: incorpora rotaciones (desplazamientos circulares de bits) cuya cantidad depende de los datos. Esto parece fortalecer el algoritmo contra el criptoanálisis.

2.7.6.1.13. Principios de criptografía de clave pública

De igual importancia que el cifrado convencional es el cifrado de clave pública, que se emplea en autenticación de mensajes y en distribución de claves.

2.7.6.1.14. Estructura del cifrado de clave pública

El cifrado de clave pública, propuesto por primera vez por Diffie y Hellman en 1976, es el primer avance realmente revolucionario en el cifrado en

miles de años. El motivo es que los algoritmos de clave pública están basados en funciones matemáticas y no en simples operaciones sobre os patrones de bits. Además, la criptografía de clave pública es asimétrica, lo que implica el uso de dos claves separadas, a diferencia del cifrado simétrico convencional, que emplea sólo una clave. El uso de dos claves tiene importantes consecuencias en el terreno de la confidencialidad, la distribución de claves y la autenticación. Antes de continuar, deberíamos mencionar algunas confusiones comunes en lo relativo al cifrado de clave pública. La primera es la creencia de que el cifrado de clave pública es más seguro ante el criptoanálisis que el cifrado convencional. De hecho, la seguridad de cualquier esquema de cifrado depende de la longitud de la clave y el coste computacional necesario para romper un cifrado. No hay nada sobre el cifrado convencional ni de clave pública que haga a uno superior al otro en lo que respecta a la resistencia al criptoanálisis. Otra equivocación la hallamos en la idea de que el cifrado de clave pública es una técnica con propósitos generales que ha dejado desfasado el cifrado convencional. Por el contrario, debido al coste computacional de los esquemas actuales de cifrado de clave pública, no parece que el cifrado convencional vaya a abandonarse. Por último, se piensa que la distribución de claves no es importante cuando se usa el cifrado de clave pública, en comparación con los incómodos acuerdos previos que tienen lugar con los centros de distribución de claves para el cifrado convencional. De hecho, se necesita alguna forma de protocolo, que a menudo implica a un agente central, y los procedimientos que tienen lugar no son más sencillos ni más eficientes que los que se requieren para el cifrado convencional.

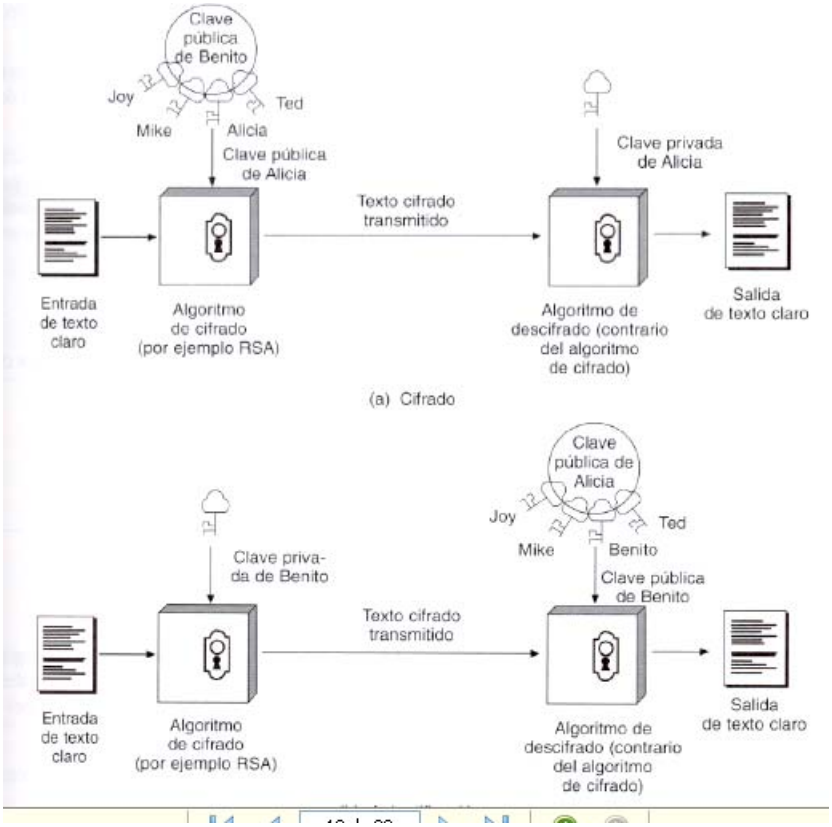
Un esquema de cifrado de clave pública tiene seis componentes:

- Texto claro: consiste en el mensaje o los datos legibles que se introducen en el algoritmo como entrada.

- Algoritmo de cifrado: el algoritmo de cifrad realiza diferentes transformaciones en el texto claro.

- Clave pública y privada: es una pareja de claves que han sido seleccionadas, de las cuales una se usa para el cifrado y la otra para el descifrado. Las transformaciones exactas llevadas a cabo por el algoritmo de cifrado dependen de la clave pública o privada que se proporciona como entrada.

- Texto cifrado: es el mensaje desordenado producido como salida. Depende del texto claro y de la clave. Para un mensaje dado, dos claves diferentes producirán dos textos cifrados diferentes. Algoritmo de descifrado: este algoritmo acepta el texto cifrado y la clave correspondiente y produce el texto claro original.



Como sugieren los nombres, la clave pública de dicha pareja de claves se hace pública para que otros la usen, mientras que la clave privada sólo es conocida por su propietario. Un algoritmo criptográfico de clave pública con propósito general se basa en una clave para el cifrado y otra diferente, aunque relacionada, para el descifrado.

Los pasos fundamentales son los siguientes:

- Cada usuario genera una pareja de claves para el cifrado y el descifrado de mensajes.

- Cada usuario localiza una de las dos claves en un registro público u otro archivo accesible. Esta es la clave pública. La otra clave no se revela. Como sugiere la figura 7ª, cada usuario mantiene un grupo de claves públicas que han obtenido de otros.

- Si Benito quiere enviar un mensaje privado a Alicia, cifra el mensaje usando la clave pública de Alicia.

- Cuando Alicia recibe el mensaje lo descifra usando su clave privada. Ningún otro receptor puede descifrar el mensaje porque sólo Alicia conoce su clave privada. En este enfoque, todos los participantes tienen acceso a las claves públicas, y las claves privadas las genera cada

participante de forma local y, por tanto, nunca necesitan ser distribuidas. Mientras el usuario proteja su clave privada, la comunicación entrante es segura. En cualquier momento un usuario puede cambiar la clave privada y publicar la clave pública que la acompaña para sustituir la clave pública antigua.

2.7.6.1.15. Aplicaciones para criptosistemas de clave pública

Los sistemas de clave pública se caracterízan por el uso de un tipo de algoritmo criptográfico con dos claves, una no se revela y la otra sí. Dependiendo de la aplicación, el emisor usa su clave privada o la clave pública del receptor, o las dos, para realizar algún tipo de función criptográfica. En términos generales, podemos clasificar el uso de criptosistemas de clave pública en tres categorías:

- Cifrado/descifrado: el emisor cifra un mensaje con la clave pública del receptor.
- Firma digital: el emisor firma un mensaje con su clave privada. Esto se consigue mediante un algoritmo criptográfico aplicado al mensaje o a un pequeño bloque de datos que es una función del mensaje
- Intercambio de llaves: dos partes cooperan para intercambiar una clave de sesión. Hay distintas posibilidades que implican la clave privada de una o de las dos partes. Algunos algoritmos son adecuados para las tres aplicaciones, mientras otros sólo se pueden usar para una o dos de ellas. La tabla 2 indica las aplicaciones que soportan los algoritmos, la tabla también inclute el Estándar de Firma Digital (DSS) y la criptografía de curva eliptica.

Algoritmo	Cifrado/descifrado	Firma digital	Intercambio de clave
RSA	Sí	Sí	Sí
Diffie-Hellman	No	No	Sí
DSS	No	Sí	No
Curva eliptica	Sí	Sí	Sí

2.7.6.1.16. Requisitos para la criptografía de clave pública

El criptosistema que se muestra en la figura 7 depende de un algoritmo criptográfico basado en dos claves relacionadas. Diffie y Hellman postularon este sistema sin demostrar que tales algoritmos existen.

Sin embargo, sí especificaron las condiciones que deben cumplir:

- Desde el punto de vista computacional, para una parte B es fácil generar una pareja de claves (clavepública K_{Ub}, clave privada K_{Rb}).

- Entérminos computacionales, para un emisor A que conozca la clave pública y el mensaje que ha de cifrarse, M, es fácil generar el texto cifrado correspondiente:

$$C = E_{KUb}(M)$$

- En términos computacionales, para un receptor B es fácil descifrar el texto cifrado resultante usando la clave privada para recuperar el mensaje original:

$$M = D_{KRb}(C) = D_{KRb}[E_{KUb}(M)]$$

- Desde el punto de vista computacional, es imposible que un openente, conociendo la clave pública, KU_b, determine la clave privada KR_b.

- Desde el punto de vista computacional, es imposible que un oponente, conociendo la clave pública, KU_b, y un texto cifrado, C, recupere el mensaje original, M. Podemos añadir un sexto requisito que, aunque útil, no es necesario para todas mlas aplicaciones de clave pública:

- Cualquiera de las dos claves relacionadas puede usarse para el cifrado, y la otra para eldescifrado.

$$M = D_{KRb}[E_{KUb}(M)] = D_{KUb}[E_{KRb}(M)]$$

2.7.6.1.17. Algoritmos de criptografía de clave pública

Los dos algoritmos de clave pública más usados son el RSA y el DiffieHellman, los cuales, se analizarán a continuación.

2.7.6.1.18. El algoritmo de cifrado de clave pública RSA

El RSA es un cifrado de bloque en el que el texto claro y el texto cifrado son enteros entre 0 y n – 1 para algún n. Para algún bloque de texto claro M y un bloque de texto cifrado C, el cifrado y el descifrado son de la siguiente forma:

$$C = M^e \bmod n$$
$$M = C^e \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

Tanto el emisor como el receptor deben conocer los valores de n y e, y sólo el receptor conoce el valor de d. este es un algoritmo de cifrado de clave pública con una clave pública de KU = {e, n} y una clave privada de KR = {d, n}. Para que este algoritmo sea satisfactorio para el cifrado de clave pública, se deben cumplir los siguientes requisitos:

- Que sea posible encontrar valores de e, d, n tal que Med = M mod n para todo M < n.

- Que sea relativamente fácil calcular Me y Cd para todos los valores de M < n.

- Que sea imposible determinar d dados e y n. Los dos primeros requisitos se cumplen fácilmente. El tercero se puede cumplir para valores grandes de e y n.

La figura siguiente resume el algoritmo RSA. Se empieza seleccionando dos números primos, p y q, y calculando su producto n, que es el módulo para cifrado y descifrado. A continuación, se necesita la cantidad Φ(n), conocida como función totient de Euler de n, que es el número de enteros positivos menor que n y primo relativo de n. Luego, se selecciona un entero e que sea primo relativo de Φ(n) [el mayor común divisor de e y Φ(n) es 1]. Finalmente, se calcula d como el inverso multiplicativo de e, módulo Φ(n). Se puede demostrar que d y e tienen las propiedades deseadas.

Generación clave

Seleccionar p, q	p y q primos, $p \neq q$
Calcular $n = p \times q$	
Calcular $\phi(n) = (p-1)(q-1)$	
Seleccionar entero e	$gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calcular d	$de \bmod \phi(n) = 1$
Clave pública	$KU = \{e, n\}$
Clave privada	$KR = \{d, n\}$

Cifrado

Texto claro:	$M < n$
Texto cifrado:	$C = M^e \pmod n$

Descifrado

Texto cifrado:	C
Texto claro:	$M = C^d \pmod n$

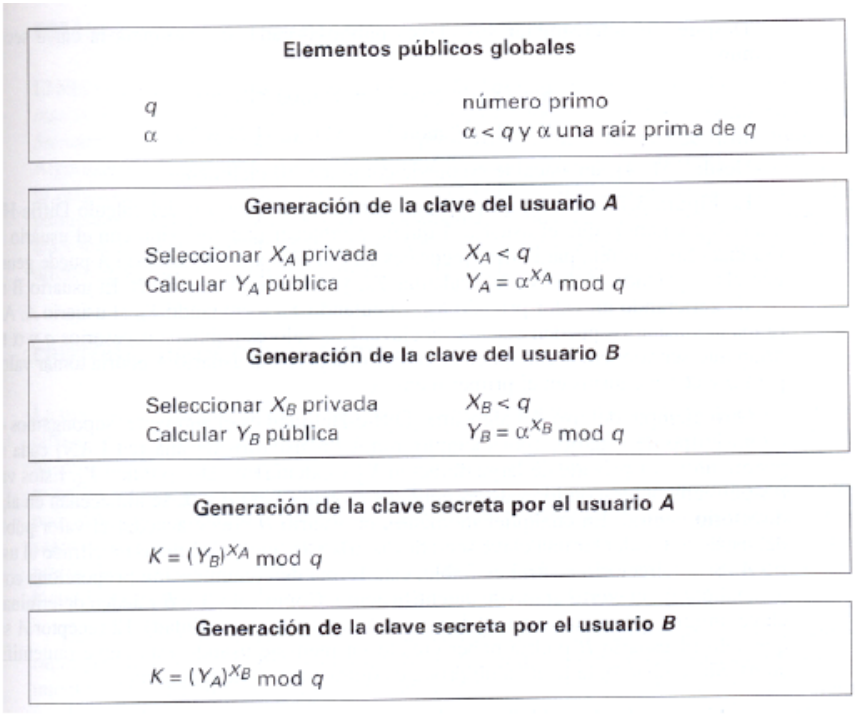
Hay dos enfoques posibles para romper el algoritmo RSA. El primero es el enfoque de fuerza bruta: intentar todas las claves privadas posibles. Así, cuanto mayor sea el número de bits en e y d, más seguro será el algoritmo. Sin embargo, debido a que los cálculos que tienen lugar tanto en la generación de la clave como en el cifrado/descifrado son complejos, cuanto mayor sea el tamaño de la clave, más lento irá el sistema. La mayoría de las discusiones sobre ciptoanálisis del RSA se han centrado en la tarea de factorizar n en sus dos factores primos. Un número n producto de dos números primos grandes es difícil factorizar, aunque no tanto como solía ser. Una ilustración llamativa de ello ocurrió en 1977; los tres inventores de RSA lectores de Scientific American a descodificar un cifrado que publicaron en la columna de juegos matemáticos de Martin Gardner. Ofrecieron una recompensa de 100 dólares por la recuperación de una frase de texto claro, algo que, según predijeron, no ocurriría durante unos 40 cuatrillones de años. En abril de 1994, un grupo que

trabajaba en Internet y que usaba unos 1.600 computadores consiguió el premio después de ocho meses de trabajo. En este reto se usó un tamaño de clave pública (longitud de n) de 129 dígitos decimales, alrededor de 428 bits. Este resultado no invalida al RSA; simplemente significa que debe usarse un tamaño de clave mayor. Actualmente, un tamaño de clave de 1024 bits (300 dígitos decimales aproximadamente) se considera lo suficientemente robusto para casi todas las aplicaciones.

2.7.6.1.19. Intercambio de clave Diffie – Hellman

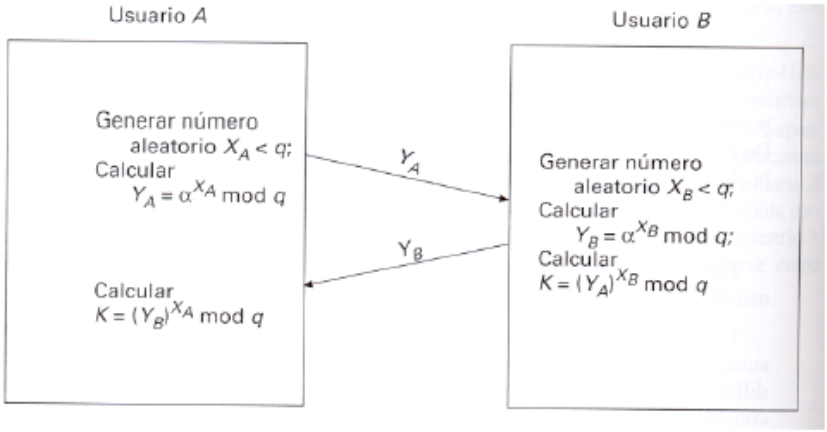
El primer algoritmo de clave pública apareció en el artículo de Diffie y Hellman que definía la criptografía de clave pública y que se conoce por el intercambio de clave Diffie – Hellman. Una serie de productos comerciales empleados emplearon ésta técnica de intercambio de claves. La finalidad del algoritmo es hacer posible que los usuarios intercambien de forma segura una clave secreta que luego pueda ser usada para el cifrado posterior de mensajes. El algoritmo está limitado al intercambio de claves.

El algoritmo de Diffie – Hellman depende para su efectividad de la dificultad de computar logaritmos discretos. En resumen, podemos definir el logaritmo discreto de la siguiente forma: primero definimos una raíz primitiva de un número primo p cuyas potencias generan todos los enteros desde 1 a p – 1. Es decir, si a es una raíz primitiva del número primo p, entonces los números a mod p, a2 mod p …, ap1 mod p son distintos y consisten en los enteros desde 1 hasta p – 1 en alguna de sus permutaciones Para cualquier entero b menor que p y una raíz primitiva a del número primo p, se puede encontrar un único exponente i tal que b = ai mod p donde 0 ≤ i ≤ (p 1) El exponente i se conoce como el logaritmo discreto o índice de b para la base a, mod p. Este valor se representa como inda,p (b). Con toda esta información se puede definir el intercambio de clave de Diffie – Hellman, que se resume en la figura siguiente. Para este esquema, hay dos números conocidos públicamente: un número primo q y un entero que es la raíz α primitiva de q.



La seguridad del intercambio de claves de Diffie – Hellman se basa en el hecho de que, aunque es relativamente fácil calcular exponenciales módulo un número primo, es muy difícil calcular logaritmos discretos. Para números primos grandes, la última tarea se considera imposible.

La figura que se muestra a continuación muestra un protocolo simple que hace uso del cálculo Diffie – Hellman. Supongamos que el usuario A quiere establecer una conexión con el usuario B y usa una clave secreta para cifrar mensajes en esa conexión. El usuario A puede generar una clave privada exclusiva XA, calcular YA, y enviarlo al usuario B. El usuario B responde generando un valor privado XB, calculando YB y enviando YB al usuario A. Ahora los dos usuarios pueden calcular la clave. Los valores públicos necesarios q y α tendrían que ser conocidos con antelación. Por otra parte, el usuario A podría tomar valores para q y α e incluirlos en el primer mensaje.



Otro ejemplo del uso del algoritmo Diffie – Hellman es el siguiente: supongamos que en un grupo de usuarios (por ejemplo, todos los usuarios de una red LAN) cada uno genera un valor privado de larga duración XA y calcula un valor público YA. Éstos valores públicos, junto con los valores públicos globales para q y , se almacenan en algún directorio central. En cualquier momento, el usuario α B puede acceder al valor público del usuario A, calcular una clave secreta y usarla para enviar un mensaje cifrado al usuario A. Si el directorio central es confiable, esta forma de comunicación proporciona confidencialidad y un cierto grado de autenticación. Como solo A y B pueden determinar la clave, ningún otro usuario puede leer el mensaje (confidencialidad). El receptor A sabe que sólo el usuario B podría haber creado un mensaje usando esta clave (autenticación). Sin embargo, la técnica no protege contra ataques de repetición.

2.7.6.1.20. Firmas Digitales

El cifrado de clave pública se puede usar de otra forma, como ilustra la figura siguiente. Supongamos que Benito quiere enviar un mensaje a Alicia y, aunque no es necesario que el mensaje se mantenga en secreto, quiere que Alice se asegure de que el mensaje, efectivamente, proviene de él. En este caso Benito usa su propia clave privada para cifrar el mensaje. Cuando Alice recibe el texto cifrado, se encuentra con que puede descifrarlo con la clave pública de Benito, demostrando así, que el mensaje ha debido ser cifrado por él. Nadie más tiene la clave privada de Benito y, por lo tanto, nadie más ha podido crear un texto cifrado que pueda ser descifrado con su clave pública. Por consiguiente, sin acceso a la clave privada de Benito, así que el mensaje queda autenticado tanto en lo que respecta a la fuente como a la integridad de los datos.

En el esquema anterior, se ha cifrado todo el mensaje, que, aunque valide el autor y los contenidos, necesita una gran cantidad de almacenamiento. También se debería almacenar una copia en texto cifrado para que el origen y el contenido se puedan verificar en caso de desacuerdo. Una forma más efectiva de lograr los mismos resultados es cifrar un pequeño

bloque de bits que sea una función de un documento. Este bloque, llamado autentificador, debe tener la propiedad por la cual es imposible cambiar el documento sin cambiar el autentificador. Si el autentificador se cifra con la clave privada del emisor, sirve como firma que verifica el origen, el contenido y la secuencia. Es importante resaltar que el proceso de cifrado que se acaba de describir no proporciona confidencialidad. Es decir, el mensaje que se está enviando es seguro contra alteraciones pero no contra escuchas, lo que es obvio en el caso de una firma basada en una parte del mensaje, porque el resto del mensaje se transmite en claro. Incluso en el caso del cifrado completo, no hay protección de confidencialidad ya que cualquier observador puede descifrar el mensaje usando la clave pública del emisor.

2.7.6.1.21. Gestión de claves

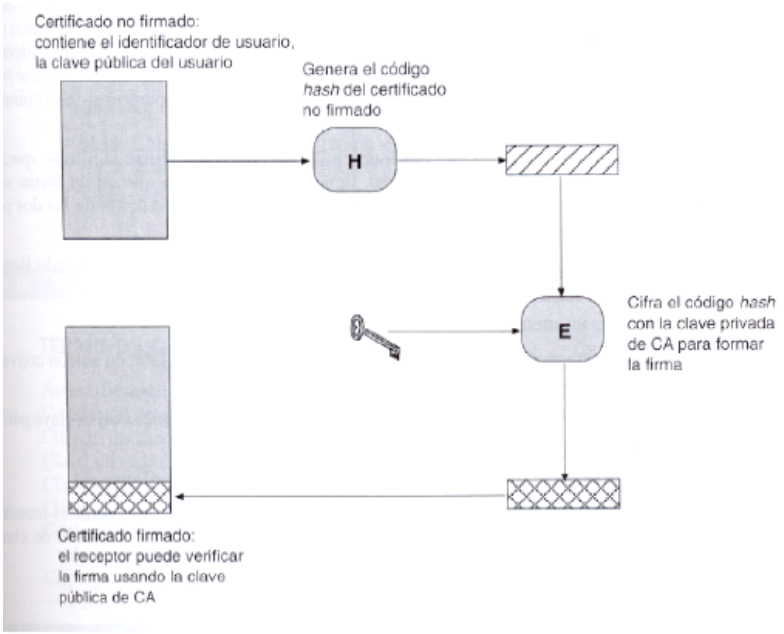
Una de las funciones principales del cifrado de clave pública es la de tratar el problema de la distribución de claves. Hay dos aspectos fundamentales sobre el uso del cifrado de clave pública en este sentido:

- La distribución de claves públicas
- El uso de cifrado de clave pública para distribuir claves secretas.

2.7.6.1.22. Certificados de clave pública

A la vista de todo esto, la base del cifrado de clave pública se encuentra en el hecho de que la clave pública es pública. Así, si hay un algoritmo de clave pública aceptado, como el RSA, cualquier participante puede enviar su clave pública a cualquier otro o difundir la clave a la comunidad en general. Aunque este enfoque es conveniente, presenta una debilidad fundamental: cualquiera puede falsificar ese dato público.

Es decir un usuario podría hacerse pasar por el usuario A y enviar una clave pública a otro participante o difundirla. Hasta el momento que A descubre la falsificación y alerta a los otros participantes, el falsificador puede leer todos los mensajes cifrados enviados a A y puede usar las claves falsificadas para la autenticación. La solución a este problema es el certificado de clave pública. Básicamente, un certificado consiste en una clave pública y un identificador o nombre de usuario del dueño de la clave, con todo el bloque firmado por una tercera parte confiable. Comúnmente, la tercera parte es una autoridad de certificación (CA, Certificate Authority) en la que confía la comunidad de usuarios, que podría ser una agencia gubernamental o una institución financiera. Un usuario puede presentar su clave pública a la autoridad de forma segura, obtener un certificado y luego publicarlo. Cualquiera que necesite la clave pública de este usuario puede obtener el certificado y verificar que es válida por medio de la firma fiable adjunta. La figura vsiguiente ilustra este proceso.



El esquema que se ha aceptado mundialmente para el formateado de los certificados de clave pública es el estándar X.509, cuyos certificados se emplean en la mayoría de las aplicaciones de seguridad de redes.

2.7.6.1.23. Distribución de claves secretas mediante criptografía de clave pública

Con el cifrado convencional, un requisito fundamental para que las dos partes se comuniquen de forma segura es que compartan la clave secreta. Supongamos que Benito quiere crear una aplicación de mensajes que le permita intercambiar correo electrónico de forma segura con alguien que tiene acceso a Internet o a otra red que ambos comparten. Supongamos además, que quiere hacerlo usando cifrado convencional. Con el cifrado convencional, Benito y su interlocutor, Alicia, deben acordar una forma de compartir una clave secreta que nadie más conozca. ¿Cómo van a hacerlo? Si Alicia se encuentra en la habitación continua a Benito, éste podría generar una clave y anotarla en un papel o guardarla en un disquete y entregarla a Alicia. Pero si Alicia esta en el otro extremo del continente o del mundo, ¿qué puede hacer Benito? Podría cifrar la clave usando cifrado convencional y enviarla por correo electrónico a Alicia, pero esto significa que ambos deben compartir una clave secreta para cifrar esta nueva clave secreta. Además, Benito y todo aquel que use este nuevo paquete de correo electrónico se enfrenta al mismo problema con cada posible interlocutor: cada pareja de interlocutores debe compartir una clave secreta única. Un enfoque consiste en el uso del intercambio de clave de Diffie. -Hellman, que, de hecho, está muy extendido. Sin embargo, tiene la desventaja de que su forma más simple el algoritmo de Diffie – Hellman no proporciona la autenticación de las dos partes que se comunican.

Una alternativa válida es el uso de los certificados de clave pública. Cuando Benito quiera comunicarse con Alicia, puede hacer lo siguiente:

- Preparar un mensaje
- Cifrar el mensaje usando cifrado convencional con una clave de sesión convencional.
- Cifrar la clave de sesión utilizando el cifrado de clave pública con la clave pública de Alicia.
- Añadir la clave de sesión cifrada al mensaje y enviarlo a Alicia. Sólo Alicia puede descifrar la clave de sesión y, por lo tanto, recuperar el mensaje original. Si Benito obtuvo la clave pública de Alicia a través del certificado de clave pública de Alicia, está seguro de que se trata de una clave válida.

2.7.7. Ejemplo de ataque: "Ataque de Inundación en Redes Ad Hoc".

2.7.7.1. Introducción

La red móvil ad hoc es un sistema autónomo de nodos móviles conectados por ligas inalámbricas. Cada nodo no sólo opera como un fin de sistema, también como un router para retransmitir los paquetes. Los nodos son libres moverse y se organizan ellos mismos en una red. Las redes móviles ad hoc no requieren una infraestructura fija tales como estaciones base, además, es una opción atractiva para tener una red de dispositivos móviles de forma rápida y espontánea, tal como aplicaciones militares, operaciones de emergencia, dispositivos de red electrónicos personales y aplicaciones civiles como un salón de clases. Las redes ad hoc móviles tienen varias características sobresalientes, como son, las topologías dinámicas, la capacidad reducida de ancho de banda, capacidad variable en las ligas, debido a estas características, las redes móviles ad hoc son particularmente vulnerables a ataques por negación de servicio lanzado por un nodo intruso.

Las redes ad hoc móviles son desplegadas a menudo en ambientes dónde los nodos son desatendidos y tienen pequeña o ninguna protección física contra manipulaciones. Los nodos de las redes móviles ad hoc son así susceptibles a estar en peligro. Las redes son particularmente vulnerables a ataques de rechazo del servicio (DOS) lanzados por intrusos. Se presentará un nuevo ataque de DOS y su defensa en las redes ad hoc. El nuevo ataque de DOS, llamado "Ataque de Inundación en Redes Ad Hoc", produce la negación de servicio cuando se están usando protocolos de ruteo bajo demanda para redes Ad Hoc, tales como AODV y DSR. El intruso transmite excesivos paquetes RREQ o envía muchos paquetes de DATOS para agotar el ancho de banda entre las comunicaciones y los recursos en los nodos, para que las comunicaciones válidas no puedan llevarse a cabo. Después de analizarse dicho ataque (ataque de inundación en redes Ad Hoc), se ha desarrollado una

defensa genérica llamada Prevención al Ataque de Inundación (Flooding Attack Prevention, FAP). El FAP está compuesto de la supresión de vecino y corte de ruta. Cuando el intruso transmite paquetes "Route Request (RREQ)", los vecinos inmediatos del intruso observan una tasa alta de RREQ enviados por el atacante, entonces ellos bajan la prioridad de acuerdo a la tasa de RREQ entrantes, en pocas palabras la prioridad de un nodo va a ser inversamente proporcional a los paquetes RREQ que envía dicho nodo. Las peticiones de prioridad baja son eventualmente descartadas. Cuando el intruso envía muchos paquetes de DATOS al nodo víctima, el nodo puede cortar el camino y no volver a aceptar una ruta con el intruso. El ataque de inundación de redes Ad Hoc puede ser evitado fácilmente con FAP.

2.7.7.2. Resumen del protocolo de ruteo AODV

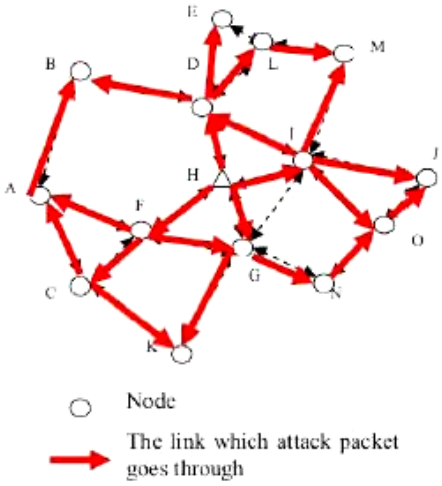
En AODV, el descubrimiento de rutas es completamente contrademanda. Cuando un nodo fuente necesita enviar paquetes a un destino al cual, no tiene ninguna ruta válida, el nodo transmite un paquete RREQ (Petición de Ruta) a sus vecinos. Cada nodo mantiene un único y creciente número de secuencia para asegurar rutas fuera de loops. El nodo fuente incluye el número de secuencia que conoce del destino en el paquete RREQ. El nodo intermedio recibe el paquete RREQ y verifica en las entradas de su tabla de ruteo si posee una ruta hacia el destino con un número de secuencia mayor que en el paquete de RREQ, si lo tuviese manda un paquete RREP (Contestación de Ruta) hacia el nodo fuente, a través de sus vecinos por los que recibió el paquete de RREQ. Por otra parte, si no tuviese un número de secuencia actual para el nodo destino, añade en su tabla de ruteo una entrada indicando que ya sabe como llegar a la fuente que generó el RREQ (camino inverso) y entonces retransmite el paquete de RREQ a sus vecinos. Los paquetes RREQ duplicados recibidos por un nodo son tirados. De esta manera, el paquete RREQ se inunda en una forma controlada en la red, y llegará eventualmente al destino mismo o a un nodo, el cual contenga una ruta actualizada en su tabla de ruteo, cualquiera de los dos generará un paquete RREP y lo mandará hacia la fuente por la misma ruta por la que le llego el RREQ. AODV también incluye el mecanismo de mantenimiento de ruta para manejar la topología dinámica de la red. Las ligas que se rompen pueden ser detectadas ya sea por beacons periódicos o acknowledgments de la capa de enlace, tal como aquéllos proporcionados por el protocolo 802.11 MAC. Una vez que una liga está rota, un no solicitado paquete RRER con un número de secuencia actual y un número de saltos finito es propagado a todos los nodos fuentes activos, los cuales estén actualmente usando dicho link. Cuando el nodo fuente recibe la notificación de que una liga está rota, puede reiniciar la ruta con el proceso de descubrimiento de ruta, esto es si aun esta interesado tener dicha ruta hacia el destino.

2.7.7.3. Ataque de inundación RREQ

La inundación de paquetes RREQ en toda la red consume muchos recursos. Para reducir la congestión en una red, el protocolo de AODV adopta algunos métodos. Un nodo no puede originar más de RREQ_RATELIMIT mensajes RREQ por segundo. Después de transmitir un RREQ, un nodo

espera por un RREP. Si para dicha ruta no se recibe el RREP dentro de "roundtrip" milisegundos, el nodo puede intentar descubrir la ruta de nuevo transmitiendo otro RREQ, a un máximo de intentos al valor de TTL máximo. Repetidos esfuerzos por un nodo fuente al descubrimiento de ruta para un solo el destino debe utilizar un "binary exponential backoff". La primera vez un nodo fuente transmite un RREQ, él espera un tiempo "roundtrip" para la recepción de un RREP. Si un RREP no se recibe dentro de ese tiempo, el nodo fuente envía un nuevo RREQ. Al calcular el tiempo de espera para el RREP después de enviar el segundo RREQ, el nodo fuente debe usar un "binary exponential backoff". Por lo tanto, el tiempo de espera por el RREP correspondiente al segundo RREQ es 2 * roundtrip time. Los paquetes RREQ son transmitidos en un anillo creciente reduciendo el overhead causado por la inundación de la red entera. Los paquetes se inundan en una área pequeña (un anillo) primero definido por un TTL inicial (tiempodevida) en la cabecera IP. Si no se ha recibido el RREP, el área inundada se agranda aumentando el TTL por un valor fijo. El procedimiento se repite hasta que un RREP se reciba por el creador del RREQ, es decir, la ruta se ha encontrado.

El Ataque de Inundación en redes ad hoc, el nodo atacante viola las reglas anteriores para agotar los recursos de la red. Primeramente, el intruso selecciona muchas direcciones IP que no están en la red, esto es, si él sabe el alcance de direcciones IP en la red. Porque ningún nodo puede responder los paquetes de RREP para éstos RREQ, la ruta inversa en la tabla de ruteo del nodo será conservado mucho tiempo. El atacante puede seleccionar al azar las direcciones IP si es que él no puede saber alcance de las direcciones IP. Después, el atacante origina masivamente mensajes RREQ para estas direcciones IP nulas. El atacante intenta enviar RREQ's sin considerar un RREQ_RATELIMIT por segundo. El atacante reenviara paquetes RREQ sin esperar por el RREP o roundtrip time. El TTL de los RREQ's se inicializan al máximo sin usar el método de expansión de anillo. En los ataques de inundación, la red entera estará llena de paquetes RREQ que el atacante envía. El ancho de banda de la comunicación es agotada por la inundación de paquetes RREQ así como los recursos en los nodos. Por ejemplo, el almacenamiento de la tabla de ruteo es limitada. Si masivos paquetes de RREQ están llegando a un nodo en poco tiempo, el almacenamiento en la tabla de ruteo del nodo se agotará y éste no podrá recibir nuevos paquetes RREQ. Como resultado, los nodos legítimos no pueden no descubrir rutas para enviar datos. La figura 1 muestras un ejemplo de un ataque por inundación de paquetes RREQ. El nodo H es el atacante e inunda con paquetes RREQ toda la red para que otros nodos no puedan construir rutas.



2.7.7.4. Ataque por inundación de datos.

Primero, el atacante crea rutas a todos los nodos en la red. Después, manda excesivamente paquetes de datos inútiles a través de dichas rutas. Los excesivos paquetes de datos en la red agotan el ancho de banda disponible para las comunicaciones entre los nodos. El nodo destino estará ocupado recibiendo los paquetes y no podrá trabajar normalmente. La característica común de dos tipos de ataques de inundación es agotar el ancho de banda disponible de la red afectando así la comunicación legítima. Por otra parte, cada ataque tiene sus características particulares. El ataque por inundación de paquetes RREQ produce desbordamiento en la tabla de ruteo del nodo para que el nodo no pueda recibir nuevos paquetes RREQ. En el ataque de inundación de paquetes de DATOS, el proceso de recibir el ataque de paquetes consumirá muchos recursos en los nodos. Si el atacante combina dos tipos de ataques, producirá el colapso de la red entera.

Comparación entre ataques de inundación en redes Ad Hoc y SYN.

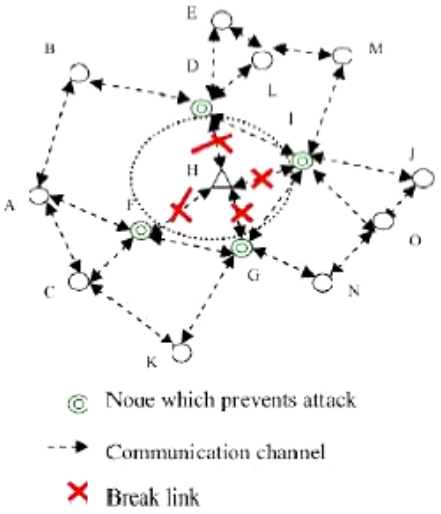
El ataque de inundación aprovecha el mecanismo de “treeway handshake” de TCP/IP y su limitación en mantener conexiones entreabiertas. Cuando un servidor recibe una petición SYN, éste regresa un paquete SYN/ACK al cliente. Hasta que el paquete SYN/ACK es reconocido por el cliente, la conexión se mantiene en un estado entre abierto por un periodo hasta que se llega a un timeout. El cual es típicamente de 75 segundos. El servidor ha construido en su sistema de memoria una cola para mantener todas las conexiones entreabiertas. Desde que esta cola es de tamaño finito, una vez que ha alcanzado su límite, todas las demás peticiones a conexión serán rechazadas. Si una petición SYN es alterada, el servidor victima nunca recibirá el paquete ACK final para completar el “treeway handshake”.

Name	SYN Flooding Attack	Ad Hoc Flooding Attack
Attack method	TCP connection requests with spoofed source addresses	Flooding mass RREQ and DATA packets
Victim	host	Mobile ad hoc networks
Protocol	TCP/IP	On-demand routing protocol
Protocol layer	Transport layer	Network layer
goal	Denial of service in host	Denial of service in the whole networks

2.7.7.5. Eliminación de vecinos

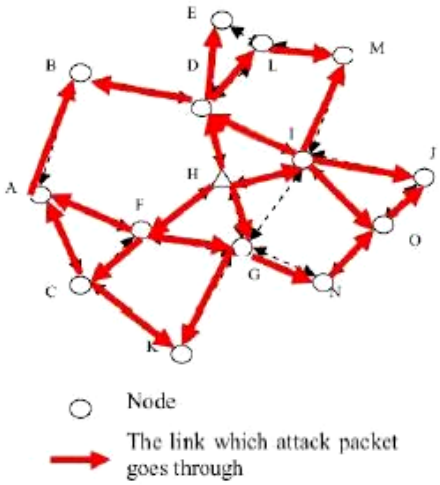
El método de eliminación de vecino es usado para prevenir el ataque por inundación de paquetes RREQ. Las redes móviles Ad Hoc son redes inalámbricas multisalto, y los nodos mandan y reciben paquetes a través de sus nodos vecinos. Si todos los nodos vecinos alrededor de un nodo se rehúsan a retransmitir sus paquetes, el nodo no se podrá comunicar con los otros nodos en la red ad hoc, aislándose así de la red.

La figura siguiente muestra una topología de red ad hoc móvil. El nodo H se comunica con los otros nodos a través de los nodos D, F, G e I. Si los nodos vecinos D, F, G e I se negasen a recibir paquetes del nodo H, el nodo H no podrá enviar ningún paquete a los otros nodos.



En AODV, los nodos colocan los paquetes RREQ de acuerdo a la regla “firstin, firstout” (FIFO). Con dicha regla, los excesivos paquetes RREQ del atacante harán que se sature la cola y los nodos no podrán recibir los paquetes RREQ posteriores. Se cambiara la cola FIFO por una cola de prioridad. Además se tendrá un umbral de paquetes que se podrán recibir por cada vecino. La prioridad de un nodo es inversamente proporcional a la frecuencia que origina RREQ. El umbral es el número máximo de paquetes RREQ que origina un nodo en un periodo determinado, tal como 1 seg. Si la frecuencia de originar RREQ del atacante excede el umbral, su vecino no recibirá más RREQ

del atacante. Así como entre mas paquetes RREQ mande menor será su prioridad en la cola. Para clarificar, tomamos el nodo F y sus vecinos A, C, H, G de la siguiente figura. El nodo F prepara el proceso de prioridad para sus vecinos A, C, H, y G. Los valores iniciales de las cuatro prioridades en el nodo F son 1's. Después del nodo A transmite dos paquetes RREQ en 1 segundo, la prioridad del nodo A cambia a 1/2. Si el nodo C origina 5 paquetes RREQ en 1 segundo, la prioridad del nodo C se cambia a 1/5. Después de esto, si el nodo A y C transmiten paquetes RREQ al mismo tiempo, el nodo F primero transmitirá el paquete RREQ del nodo A porque la prioridad de A es más alta que de C. Si el nodo H (intruso) transmite excesivamente paquetes RREQ en un período de tiempo, la prioridad de H será muy baja. Si la frecuencia excede el umbral, nodo F negará la retransmisión de los paquetes RREQ de H, esto es semejante, para el nodo D, I, G. Como resultado, será cancelado el ataque por inundación de RREQ por medio de los vecinos.



2.7.7.6. Corte de ruta.

Cuando el intruso origina un ataque de inundación de datos; para los nodos vecinos es difícil identificarlo, ya que no pueden juzgar que un paquete de datos, es inútil en la red. Pero el nodo destino puede fácilmente determinar en la capa de aplicación cuando recibe un paquete de datos inútil. Se presentará un método llamado “corte de ruta” para prevenir el ataque por inundación de paquetes de datos. Cuando un atacante origina dicho ataque, éste encuentra un camino hacia la victima. Cuando la victima se da cuenta que está siendo atacado, él puede cortar la ruta, originando un mensaje RRER dirigido al atacante. Los nodos intermedios por los que pasa el RRER borraran la ruta del atacante hacia la victima. El mensaje RERR podría quitar algunas rutas, las cuales, no están relacionadas con el ataque, estas rutas pueden ser reparadas por los nodos en el futuro. Así las rutas se van cortando y el ataque es gradualmente terminado. Cuando el ataque es terminado, el atacante puede originar un nuevo RREQ, y el nodo victima puede rehusarse a responderlo, no contestando con el RREP. Pero ya que en el protocolo AODV los nodos intermedios pueden responder los RREQ si tienen rutas válidas aunque la

victima no quiera que la ruta se reactive. Para evitar esto, la función de que los nodos intermedios puedan contestar RREQ debe ser cancelada. Solamente el destino puede responder los RREQ.

2.7.7.7. Conclusiones.

Se ha descrito el ataque por inundación, un ataque poderoso contra los protocolos de ruteo bajodemanda para redes ad hoc. Este ataque permite al intruso montar un ataque de rechazo de servicio contra todos los protocolos de ruteo que trabajan bajodemanda para redes móviles ad hoc, incluso los protocolos seguros. Se ha diseñado el protocolo llamado “Prevención del Ataque de Inundación” (FAP) para resistir el ataque. El FAP está compuesto de dos técnicas, qué son la supresión de vecino y corte de ruta, y estas técnicas son capaces de defender el Ataque Inundando eficazmente

3. Enrutamiento

3.1. Introducción al enrutamiento.

3.2. Protocolos Proactivos:

3.2.1. OLSR: Optimized Link State Routing Protocol

3.2.1.1. Visión general:

OLSR fue concebido como una adaptación de protocolos de redes no inalámbricas, más concretamente del RIP y del OSPF. Este protocolo se clasifica dentro del conjunto de protocolos proactivos, es decir, cada nodo intercambia información regularmente con otros nodos de la red. Por tanto un nodo, no solo requiere intercambiar información, si no también almacenarla, para así poder compartir la información que posee de la red con otros nodos de la misma. Cada nodo posee un grafo de interconexión con otros nodos. Nótese, que no es un grafo total de información de toda la red, cada nodo almacenará información parcial de la misma.

El tipo de enrutamiento es *"hop-by-hop",* es decir, cuando un nodo recibe un paquete y usa la información local que tiene para reenviarlo a otros nodos. Esta información se almacena en distintas tablas de tuplas con distintas direcciones y datos útiles que la red utiliza para el enrutamiento.

Cada nodo selecciona de sus vecinos un conjunto de nodos que denominamos *"MultiPoint Relays" (MPR)* estos nodos son los encargados de retransmitir y controlar el trafico de la red de sus vecinos al resto de la red. Antes de entrar en profundidad en este apartado destacaremos también que estos MPR deben de ser nodos bidireccionales. Gracias a este sistema de enrutamiento mediante los MPR se reduce el número de retransmisiones requeridas para enviar "inundar" un mensaje en toda la red, esto reduce la sobrecarga de la red y, por tanto, reduce las colisiones entre los paquetes. No es obligatorio ningún nº específico de MPRs pero si lo es para poder garantizar el camino más corto entre dos nodos. Realmente, el único requerimiento es que cada MPR retransmita a sus vecinos el estado del resto de vecinos alcanzables a través de dicho MPR.

Este protocolo es más adecuado para redes con transmisions aleatorias y de configuraciones esporádicas que para redes fijas. Es decir, es una buena solución en redes de alta transmisión, ya que no genera transmisiones adicionales en caso de caida o de movimiento de un nodo.

Al ser una red proactiva, como hemos dicho, tiene que mantener información del estado de la red, para ello es necesario que cada cierto tiempo mande paquetes de control, bien para detectar los vecinos de alrededor bien para compartir información con ellos. Pero esto conlleva a un problema o duda, y es que se necesita configurar adecuadamente el tiempo de envío entre estos paquetes de control. Dicho de forma más explicita, si se establece un intervalo

pequeño de envíos se generan muchos paquetes y por tanto aumenta la sobrecarga y las colisiones de la red. Sin embargo, si el tiempo es excesivamente grande puede que la información contenida en cada nodo en el momento de retransmisión de un paquete sea obsoleta. Sería óptimo que este tiempo fuera ajustable en tiempo real, o incluso adaptativo a las características de la red. Por ejemplo en una red con alta movilidad será mejor un tiempo bajo para tener la máxima información en todo momento. Pero, no siempre es así, y no es fácil determinar un algoritmo de ajuste de tiempo.

Resaltamos que es un protocolo robusto, ya que admite una perdida razonable de paquetes de control sin que perjudique a ninguna de las partes del protocolo. Finalmente añadir que en los paquetes de datos generalmente se le añade un número de secuencia que permite que los paquetes se reciban en desorden. Esto a priori, puede parecer poco importante, pero otros protocolos requieren que los paquetes lleguen en orden, y en caso de fallo de un paquete puede producir un retraso considerable ya que no seguir recibiendo paquetes de una secuencia hasta que no recibiera el que ha dado el fallo. Pero todo estas explicaciones se pueden ver más claramente en el control de paquetes ARQ y sus distintos modos (con espera, continuo selectivo, etc) y tan solo resaltar que es útil, porque siendo más probable la colisión en un medio compartido y siendo más lento es mejor reenviar solo el paquete fallado y aceptar los paquetes en orden que cualquier otra solución que signifique perder más tiempo en los envíos.

3.2.1.2. Multi-Point Relays: MPR

Hemos comentado que existe una serie de nodos bidireccionales que van a ser considerados como nodos especiales, nodos elegidos de entre los vecinos encargados de reducir las sobrecargas de las retransmisiones. Esto se consigue gracias a que reduce las transmisiones redundantes en una misma región.

Cada nodo selecciona su conjunto MPR a un salto simétrico de distancia. Los mensajes que deseen enviar se retransmitirán a través de este conjunto MPR. Un nodo N que desea retransmitir un mensaje puede hacerlo a un MPR del conjunto de MPRs que ha seleccionado previamente, o bien, puede enviar el mensaje a uno de sus vecinos. Si escoge la primera opción, el nodo MPR que reciba el mensaje puede procesar ese mensaje y reenviarlo hacia otro MPR o bien al nodo destino. Si, por el contrario, el nodo N mandase el paquete a uno de sus vecinos, éstos podrían recibir y procesar el mensaje, pero, en ningún caso, reenviar el mensaje a cualquier otro nodo. Es decir, los únicos nodos capaces de realizar broadcast de paquetes al resto de la red son los MPR. Si un MPR que ha recibido un paquete envía ese paquete a un nodo no MPR significa que está mandando el paquete al nodo destino.

Ya hemos comentado que todo nodo MPR debe de ser bidireccional. Esto evita los problemas típicos en la comunicación a través de un nodo unidireccional.

Cuanto más pequeño sea el conjunto de nodos MPR más pequeña será la sobrecarga por envío de paquetes de control. Estos paquetes de control se envían mediante mensajes de tipo HELLO. Entre otras cosas son usados para mantener el conjunto de vecinos de MPR.

3.2.1.3. Formatos de paquete y reenvío.

OLSR comunica utilizando un formato unificado de paquetes. Esto hace más escalable el protocolo y ofrece una forma fácil de enviar diferentes tipos de información en una sola transmisión. Cada paquete encapsulará uno o varios mensajes. Estos mensajes compartirán un formato único de cabecera que permitirá a los nodos aceptar (o no) y retransmitir (o no) correctamente mensajes de tipo desconocido para ellos.

Los mensajes pueden ser enviados a todos los nodos de la red, o bien, el envío se puede limitar a nodos dentro de un diámetro (determinado por el número de saltos) desde el nodo que origina el mensaje.Han de controlarse las retransmisiones duplicadas de paquetes de control OLSR. Se controlarán localmente: cada nodo mantendrá un conjunto de duplicados para prevenir el envío de un paquete dos veces.

Un nodo puede examinar la cabecera de un mensaje para obtener la información de la distancia (en términos del número de saltos) hasta el nodo que originó el mensaje. Esto puede ser útil en situaciones en las que por ejemplo, el tiempo de información de un paquete de control recibido y almacenado en un nodo depende de la distancia al nodo origen.

3.2.1.3.1. Formato de paquete

.

Cabecera del paquete

- Tamaño del paquete (tamaño en bytes del paquete).
- Numero de secuencia del paquete (PSN). Debe ser incrementado en 1 cada vez que se transmite un nuevo paquete OLSR. Viene a ser el número de paquetes que hay circulando por la red.

.

Cabecera del mensaje

- Tipo de mensaje. Indica el tipo de mensaje que se encontrará en la parte “mensaje”
- VTime. Indica el tiempo durante el cual la información que contiene el mensaje es válida, desde que se recibe dicho mensaje.
- Tamaño del mensaje. Tamaño en bytes del mensaje, medido desde el campo tipo de mensaje del mensaje actual, hasta el mismo campo del siguiente mensaje (dentro del mismo paquete, obviamente).
- Dirección del nodo que originó el mensaje.
- Time to live (T2L): Número máximo de saltos por los que el mensaje puede ser transmitido. Antes de que el mensaje vuelva a ser retransmitido, este valor debe decrementarse en 1.

- Cuenta de Saltos. Número de saltos (o de nodos) por los que ha pasado el mensaje. Antes de ser reenviado el mensaje, se debe incrementar este valor en 1.

Número de secuencia del mensaje. Cuando un nodo crea un nuevo mensaje, asigna un identificador único al mismo y lo almacena en este campo. El valor almacenado en principio, será incrementado en 1 cada vez que el mismo nodo cree un nuevo mensaje. Viene a ser el número de mensajes que crea un nodo.

3.2.1.3.2. Procesamiento de paquetes y envío de mensajes

Cuando recibe un paquete básico, un nodo examina cada cabecera de mensaje. Basándose en el valor del campo tipo de mensaje, el nodo puede determinar el destino de dicho mensaje.

Un nodo puede recibir el mismo mensaje más de una vez, pero debe evitar el re-procesamiento de cualquier mensaje. Para ello, como dijimos antes, cada nodo mantiene un conjunto de duplicados. Este conjunto almacena información en forma de tuplas sobre los mensajes recibidos más recientemente. Esta tupla cuenta con 5 campos:

- D_addr: dirección del nodo que envió el mensaje.
- D_seq_num: número de secuencia del mensaje.
- D_iface_list: lista de direcciones a las que se retransmitió el mensaje.
- D_retransmited: valor booleano que determina si el mensaje fue retransmitido o no.
- D_time. Tiempo de expiración de la tupla, es decir, tiempo durante el que la información de la tupla es válida. Cuando este tiempo se agota, se debe eliminar la tupla del conjunto de duplicados.

Tras recibir un paquete, un nodo debe (para cada mensaje encapsulado):

1 Descartar el paquete, si éste no contiene mensajes (tamaño del mensaje <= talla de la cabecera).

2 Eliminar el mensaje si T2L <=0, es decir, si el mensaje no puede dar más saltos a través de la red.

3 Condición de procesamiento:
3.1 Si existe una entrada en el conjunto de duplicados, donde D_addr== Dirección del nodo que creó el mensaje (que estamos intentando procesar) && D_seq_num== número de secuencia del mensaje (que estamos intentando procesar), el mensaje ya habrá sido procesado y no debe volverse a procesar.
3.2 En otro caso, si el nodo es capaz de implementar lo que pide el campo tipo de mensaje, éste se procesa.

4 Condición de reenvío

4.1 En el mismo caso que 3.1, el mensaje ya habrá sido reenviado, y NO DEBERÏA reenviarse otra vez.
4.2 En otro caso:
4.2.1 Si el nodo puede procesar el tipo de mensaje, el mensaje DEBE ser considerado para ser reenviado, de acuerdo a las especificaciones dadas por el campo tipo de mensaje.
4.2.2 Si el nodo no puede procesarlo, el mensaje DEBERÍA ser considerado para ser reenviado, de acuerdo al algoritmo de reenvío por defecto, descrito a continuación.

3.2.1.3.3. Algoritmo de reenvío por defecto.

1. Si no se puede detectar al emisor del mensaje a sólo un salto del nodo que quiere volver a enviar el mensaje, el algoritmo se detiene y el mensaje no es reenviado.
2. Si en el conjunto de duplicados del nodo que quiere reenviar el mensaje existe una entrada donde D_addr== Dirección del nodo que creó el mensaje (que estamos intentando procesar) && D_seq_num== número de secuencia del mensaje (que estamos intentando procesar), el mensaje sólo será considerado para el reenvío si y sólo sí en esa misma tupla, D_retransmited es falso y la dirección del nodo que recibirá el mensaje no está entre las direcciones de D_iface_list.
3. En otro caso, si no existen entradas en el conjunto de duplicados como la descrita anteriormente, el mensaje se podrá reenviar.

Si después de estos pasos, el mensaje no puede ser reenviado, el algoritmo habrá parado. Si no, el protocolo actúa de ésta forma:
4. Si la dirección del emisor, es la dirección de un selector MPR y el T2L del mensaje es mayor o igual que 1, el mensaje debe ser transmitido.
5. Si en el conjunto de duplicados, existe una entrada con la misma dirección del nodo que originó el mensaje, y el mismo número de secuencia del mensaje, la entrada en el conjunto se actualiza como sigue:
D_time: tiempo actual.
La dirección del receptor se añade a D_iface_list
D_retransmitted toma valor True sí y sólo sí, el mensaje se retransmite cumpliendo las condiciones del punto 4.
Sí, y sólo si , de acuerdo con el punto 4, el mensaje debe ser retransmitido:
6. El T2L del mensaje se decrementa.
7. La cuenta de saltos del mensaje se incrementa.
8. Se hace un broadcast del mensaje.

3.2.1.4. Repositorios de información.

A través del intercambio de mensajes de control, cada nodo acumula información sobre la red, que se almacenará como vamos a ver:

- Base de información de asociación de múltiples nodos.

- Cada nodo de la red guarda tuplas de asociación de nodos, con estos campos:
  - l_iface_main: dirección del nodo en cuestión
  - l_iface_addr: dirección del nodo con el que se asocia.
  - l_time: Tiempo durante el que la información de la tupla es válida.

- Detección de enlaces: La base local de información de enlaces guarda información sólo de enlaces a nodos vecinos (nodos que están a sólo un salto).
  - Conjunto de enlaces. Un nodo almacena tuplas de enlaces con información de los mismos (dirección del nodo en cuestión, dirección del nodo destino, direcciones de nodos vecinos,etc…). El conjunto de estas tuplas es el conjunto de enlaces.

- Detección de vecinos: Base de información de Vecinos. La base de información de vecinos almacena información tanto sobre vecinos (direcciones de vecinos, etc.), como sobre vecinos a dos saltos (direcciones, tanto de vecinos como de vecinos a dos saltos), como de MPRs (direcciones de vecinos que son marcados como MPRs) y selectores MPR (direcciones de los vecinos que tienen marcado al nodo en cuestión como un MPR).

- Base de información sobre la topología. Cada nodo de la red mantiene información acerca de la propia red. Esta información se utilizará para realizar cálculos que permitan crear la tabla de enrutamiento.

3.2.1.5. Link Sensing:

Su función es rellenar la información local del estado de los enlaces de cada nodo mediante el intercambio de paquetes HELLO. Intercambia información con sus vecinos para formar un conjunto de nodos. En cada entrada de dicho conjunto se guarda el estado del enlace entre dos nodos mediante una tupla: (dirección propia, dirección remota)

Además, cada nodo guarda información sobre el estado del enlace con cada uno de sus vecinos, indicando si es simétrico o asimétrico. Simétrico expresa y afirma que el enlace es bidireccional. Por el contrario, asimétrico no puede afirmar que no sea bidireccional ya que simplemente puede haber escuchado una transmisión, pero no haber enviado un mensaje. Esta tabla la vamos a conocer como *Link Set*.

3.2.1.6. Procesamiento de mensajes HELLO

Un nodo A cualquiera, emite un mensaje Hello con su dirección. La llamaremos la dirección origen. Un vecino B, de este nodo A, recibe el mensaje y actualiza la tabla de información. Es importante destacar que el vecino no

retransmite ni almacena este mensaje en ninguna parte, tan solo lo procesa, actualiza el campo vTime del paquete con el valor "*validity time".*

El nodo B actualiza el conjunto de nodos de la siguiente manera: Primero comprueba si existe una tupla en la que la dirección de la interfaz de un vecino sea igual que la *dirección origen* del paquete HELLO recibido, es decir si:

L_neighbor_iface_addr == Dirección origen (A).

Si no existe, se crea una tupla con los siguientes valores:

L_neighbor_iface_addr = Dirección origen (B).
L_local_iface_addr = Dirección de la interfaz local (B)
L_SYM_time = current time - 1 (expired)
L_time = current time + validity time

Si la tupla existe se actualiza:

L_ASYM_time = current time + validity time.

3.2.1.7. Detección de vecinos:

En este caso se rellena una tabla con la información de todos los vecinos de un nodo, también mediante el envío de mensajes HELLO. En este caso se guardan la información del link sensing y además se actualiza la información a medida que se detectan cambios. Esta tabla se conoce como Neighborhood Set. A priori, podemos pensar que esta tabla almacena lo mismo que la tabla *Link Set*, pero mientras el segundo guarda la información sobre el estado de los enlaces, el segundo guarda información sobre los vecinos.

- Un nodo al recibir el paquete *Hello* crea la tupla con la dirección del emisor y comprueba el campo de *willingnes* (voluntad) Si la dirección origen del Hello está en una tupla del Neighborhood Set:

N_willingness = willingness from the HELLO message.

- Cada vez que se detecta un enlace, es decir, se crea una nueva entrada en el *Link Set*, entonces, se crea la tupla asociada a ese enlace solo que en el *Neighborhood set*:

N_neighbor_main_addr = main address of L_neighbor_iface_addr (Dirección de la tupla de link set)

- De la misma forma, cuando se cambia información en el *link* set se actualiza el status del Neighborhood *set* :

Si el vecino tiene asociada una tupla en el L.S e indica que es simétrica:
N_status = SYM
Si no…
N_status = NOT_SYM

- Cada vez que una tupla del *Link Set* es borrada, se borra también del *Neighborhood set*

3.2.1.8. Rellenado de conjuntos

3.2.1.8.1. *Rellenado 2-hop:*

Es necesario también almacenar una tabla que describa el conjunto de nodos que tiene un enlace simétrico a un vecino simétrico. De nuevo, esta tabla se rellena mediante el intercambio de mensajes periódicos de tipo *Hello*. Se procesa de la siguiente forma:

- Para cada dirección listada en el Hello con Neighbor Type igual a SYM_NEIGH o MPR_NEIGH. Si la dirección del vecino es igual a la del nodo que recibe el Hello se descartan ambos.

  En caso contrario,
  N_neighbor_main_addr = Originator Address;
  N_2hop_addr = main address of the 2-hop neighbor;
  N_time = current time + validity time.

  Esta tupla recién insertada reemplaza a aquellas que tengan la misma dirección que algún vecino y misma dirección 2-hop.

  Para cada dirección listada en el Hello con Neighbor Type igual a NOT_NEIGH se borran las tuplas:
  N_neighbor_main_addr == Originator Address
  N_2hop_addr == main address of the 2-hop neighbor

3.2.1.8.2. Rellenado del MPR Set:

Cada nodo selecciona su MPR Set de entre sus vecinos a un salto, de esta forma, todo nodo puede alcanzar a todos sus vecinos a 2 saltos del conjunto MPR. Ya hemos dicho que no hace falta que sea un conjunto mínimo de nodos, pero bien es cierto que esto reducirá las sobrecargas de las retransmisiones. Cada vez que se detecta un cambio en la topología de la red habrá que calcular el conjunto MPR.

1. MPR Set → VACIO
2. Mientras haya nodos a 2 saltos no cubiertos por el MPR actual:
2.1 Para cada vecino que no pertenece al MPR Set calculo el nº de vecinos a dos saltos, no cubiertos por el MPR Set, alcanzable desde este nodo.
Selecciona como MPR al nodo con el máximo nº calculado en 2.1

3.2.1.8.3. Rellenado del MPR Selector Set:

De la misma forma que un nodo mantiene información del conjunto de sus MPR, un MPR ha de guardar información sobre el conjunto de sus selectores. Es decir, guardar aquellos nodos que le han escogido como MPR. Para ello,

cuando recibe un mensaje Hello mira la lista de direcciones de los vecinos, y si encuentra su dirección acompañada del indicador de tipo MPR significa que ese nodo que ha enviado el Hello lo ha escogido como parte del conjunto de MPR. Por tanto, guardaría la información en la tabla:

1. Si no existe una tupla de un MPR Selector con MS_main_addr == Originator Address

Entonces se crea una tupla con:

MS_main_addr = Originator Address

2. La tupla con MS_main_addr == Originator Address

Se modifica de la siguiente forma

MS_time = current time + validity time.

3.2.1.9. Descubrimiento de topología.

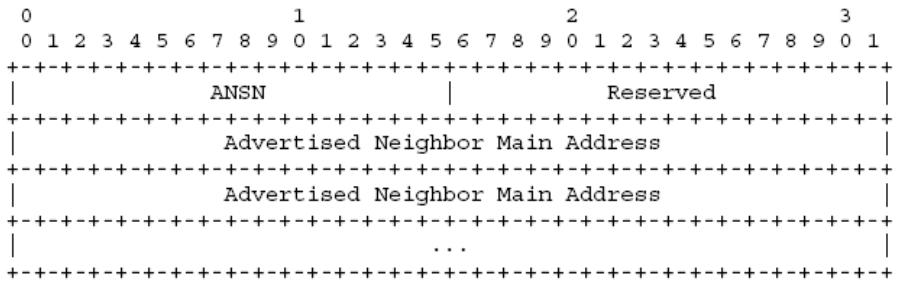
El link sensing y la detección de vecinos, permiten ofrecen a cada nodo una lista de vecinos con los cuales pueden comunicarse directamente y, en combinación con el formato de paquetes y la parte de reenvío (forwarding), un mecanismo optimizado de envío de paquetes a través de MPRs. Basándonos en esto, la información de topología, se disemina a través de la red. En esta sección describimos qué parte de la información obtenida en el link sensing y en la detección de vecinos debe ser extendida a través de toda la red, y cómo se utiliza esta información para construir rutas.

Las rutas se construyen con enlaces anunciados y enlaces con vecinos. Un nodo debe expandir información sobre sus enlaces con nodos de su conjunto de selectores MPR, para proveer suficiente información que permita el enrutamiento.

3.2.1.10. Mensajes TC

3.2.1.10.1. Formato de mensajes TC

El formato de un mensaje TC es el siguiente:



Estos mensajes son mandados en la parte de datos del formato general del mensajes (ver punto 3), cambiando el valor del campo “tipo de mensaje” a TC_MESSAGE. El T2L debería ser cambiado a 255 (el máximo valor), para difundir el mensaje por toda la red.

- Número de secuencia de un vecino anunciado. (ANSN):Se asocia un número de secuencia a cada conjunto de vecinos anunciados. Cuándo el conjunto de vecinos anunciados de un nodo varía, incrementa su ANSN. Este valor se envía en el campo "ANSN" del formato de paquete TC, para seguir la pista de la información más reciente. Cuando un nodo recibe un mensaje TC, puede decidir, basándose en su ANSN si la información recibida en el mensaje TC es o no más reciente que la que él tenía.

- Dirección principal de un vecino anunciado. Este campo contiene la dirección principal de un nodo vecino. En los mensajes TC se deben incluir todas las direcciones de nodos vecinos anunciados. Si la longitud máxima de un mensaje (impuesta por la red) no lo permite, se mandarán tantos mensajes TC como sean necesarios para, hasta que se envíen todas las direcciones de nodos vecinos.

- Reservado: Este campo es reservado y siempre tiene valor "0000000000000000"

3.2.1.10.2. Conjunto de vecinos anunciados

Un nodo envía un mensaje TC para declarar un conjunto de enlaces, llamado conjunto de enlaces anunciados, que debe incluir al menos los enlaces a todos los nodos de su conjunto de selectores MPR.

El número de secuencia (ANSN) asociado al conjunto de vecinos asociados, también es enviado con la lista de direcciones. El ANSN debe incrementarse cuando se eliminan entradas del conjunto de vecinos anunciados. Además, el ANSN debería incrementarse cuando se añaden nuevas entradas al conjunto de vecinos anunciados.

3.2.1.10.3. Generación de mensajes TC.

Cada nodo marcado como MPR, hace broadcasting de mensajes de control de topología (TC) con el fin de construir la base de información de topología.

La lista de direcciones puede ser parcial en cada mensaje TC (por ejemplo, debido a las limitaciones de tamaño de mensaje, impuesto por la red), pero tras haber recibido todos los mensajes TC describiendo un conjunto de enlaces avisados, la lista de vecinos avisados debe estar completa. La información difundida por la red de estos mensajes TC permitirá a cada nodo calcular su tabla de enrutamento.

Cuando el conjunto de enlaces anunciados de un nodo se vacía, el nodo debe continuar enviando mensajes TC (vacíos) durante un tiempo igual al "tiempo de validez" de los mensajes TC que había enviado previamente

(cuando su conjunto de enlaces anunciados aún tenía entradas), con el fin de invalidarlos. Después debería dejar de enviar mensajes TC hasta que algún nodo sea insertado en su conjunto de nodos anunciados.

Un nodo debe transmitir mensajes TC adicionales para incrementar su reactividad ante fallos en enlaces. Cuando se produce un cambio en el conjunto de selectores MPR, y ese cambio puede ser atribuido al fallo de un enlace, se debería transmitir un nuevo mensaje TC.

3.2.1.10.4. Reenvío de mensajes TC

Los mensajes TC se reenvían y se transmiten por los MPRs siguiendo el algoritmo de reenvío por defecto (descrito en la sección 3.4).

3.2.1.10.5. Procesamiento de mensajes TC

Una vez recibido un mensaje TC, se extrae su “tiempo de validez” del campo Vtime de la cabecera del paquete (ver sección 3). El conjunto de topología debería entonces actualizarse así:

1. Si la dirección del emisor (que no “originador”) del mensaje no está en la lista de vecinos de este nodo, el mensaje debe ser descartado.
2. Si existe alguna tupla en el conjunto de topología donde:
T_last_addr == dirección del nodo creador del mensaje (“originador”) AND
T_seq > ANSN
No se debe realizar el procesamiento de este mensaje y debe ser descartado debido a que el mensaje ha sido recibido fuera de orden.
3. Todas las tuplas en el conjunto de topología con:
T_last_addr == dirección del nodo creador del mensaje (“originador”) AND
T_seq < ANSN
Deben ser eliminadas del conjunto de topología.
4. Para cada dirección de vecino anunciado recibida en el mensaje TC:
4.1 Si existe alguna tupla en el conjunto de topología que cumpla:

T_dest_addr == dirección de un nodo vecino AND
T_last_addr == dirección del “originador”
Entonces el tiempo de validez de esta tupla se debe actualizar a
T_time = tiempo actual + tiempo de validez.
4.2 En otro caso se debe crear una nueva entrada en el conjunto de topología, con los valores:
T_dest_addr = dirección del nodo vecino.
T_last_addr = dirección del originador
T_seq = ANSN
T_time = tiempo actual + tiempo de validez.

3.2.1.10.6. Cálculo de la Tabla de enrutamiento.

Cada nodo mantiene una tabla de enrutamiento que le permite enrutar datos destinados a otros nodos de la red. La tabla de enrutamiento se basa en la información contenida en la base de información de enlaces local y en el conjunto de topología. Sin embargo, si alguno de estos conjuntos cambia, la tabla de enrutamiento debe ser recalculada para actualizar la información de ruta a cada destino de la red. Las entradas de ruta se almacenan en la tabla de enrutamiento con este formato:

1. R_dest_addr R_next_addr R_dist R_iface_addr
2. R_dest_addr R_next_addr R_dist R_iface_addr
3. R_dest_addr R_next_addr R_dist R_iface_addr

…

Cada entrada en la tabla estima que el nodo identificado con R_dest_addr se encuentra a una distancia R_dist saltos del nodo local, que el nodo vecino con dirección R_next_addr es el siguiente salto en la ruta hacia R_dest_addr, y que este nodo es alcanzable desde los nodos con direcciones incluidas en R_iface_addr . Las entradas se almacenan para cada destino en la red para los que se conoce una ruta. Todos los destinos para los que la ruta está rota o es parcialmente conocida, no se incluyen en la tabla.

La tabla de enrutamiento es actualizada cuando se produce un cambio en uno de éstos conjuntos: conjunto de enlaces, conjunto de vecinos, conjunto de vecinos a dos saltos, conjunto de topología o base de información de asociaciones de múltiples interfaces.

La tabla de enrutamiento se recalcula cuando se produce una aparición o desaparición de un vecino, cuando se crea o se elimina una tupla de 2 saltos, o cuando se crea o se elimina una tupla en el conjunto de topología. La actualización de esta información de enrutamiento no provoca la creación ni el envío automático de nuevos paquetes.

Para construir la tabla de enrutamiento de un nodo X, se ejecuta en algoritmo de búsqueda del camino más corto sobre el grafo directo que contiene los arcos X → Y donde Y es un nodo vecino simétrico de X (tipo de vecino = SYM), los arcos Y→Z donde Y es un nodo vecino con “willignes” distinto de WILL_NEVER y existe una entrada en el conjunto de vecinos a dos saltos con Y como N_neighbor_main_addr y Z como N_2hop_addr, y los arcos U→V donde existe una entrada en el conjunto de topología tal que V es T_dest_addr y U es T_last_addr.

Procedimiento dado como ejemplo de cómo calcular (o recalcular) la tabla de enrutamiento:

1. Todas las entradas existentes en la tabla se eliminan
2. Las nuevas entradas se añaden empezando por los vecinos simétricos como nodos destino. Así, para cada tupla del conjunto de vecinos donde N_status=SYM (hay un enlace simétrico al vecino), y por cada tupla de enlace asociado al nodo vecino, se almacena una nueva entrada en la tabla con:
R_dest_addr = L_neighbor_iface_addr, de la tupla de links asociados.
R_next_addr = L_neighbor_iface_addr, de la tupla de links asociados.

R_dist = 1;
R_iface_addr= L_local_iface_addr de la tupla de links asociados.

Si ningún R_dest_addr es igual a la dirección del vecino, se debe crear una nueva entrada a la tabla de enrutamiento con:
R_dest_addr = dirección del vecino.
R_next_addr = L_neighbor_iface_addr, de la tupla de links asociados.
R_dist = 1;
R_iface_addr= L_local_iface_addr de la tupla de links asociados.

3. Para cada nodo en N2, (vecinos a 2 saltos, para los que existe por lo menos una entrada en el conjunto de vecinos a 2 saltos donde N_neighbor_main_addr corresponde a un nodo vecino con "willingness" distinto de WILL_NEVER, uno selecciona una de esas tuplas y crea una nueva entrada en la tabla de rutas con:
R_dest_addr = dirección del vecino a 2 saltos.
R_next_addr = R_next_addr, de la entrada en la tabla de enrutamiento con: R_dest_addr == N_neighbor_main_addr de la tupla del conjunto de vecinos a 2 saltos.
R_dist = 2;
R_iface_addr= R_iface_addr de la tupla en la tabla de enrutamiento con: R_dest_addr == N_neighbor_main_addr de la tupla del conjunto de vecinos a 2 saltos.

3.2.1.11. Configuración de nodos.

Cómo debería ser configurado un nodo para trabajar en una MANET con OLSR:

3.2.1.11.1. Asignación de direcciones

Las direcciones se deberían asignar a los nodos de una MANET de entre una secuencia definida de direcciones, es decir, los nodos en la MANET deberían ser direccionables a través de una dirección de red y de una máscara de red. De la misma forma, a los nodos de cada red asociada se les deberían asignar direcciones de entre una secuencia de direcciones definidas, distinta de la utilizada en la MANET.

3.2.1.11.2. Configuración de enrutamiento

Algún nodo de la MANET con redes o hosts asociados debería ser configurado de tal forma que tuviera almacenadas rutas a todas las interfaces con la red asociada o los hosts asociados.

3.2.1.11.3. Reenvío de paquetes de datos.

OLSR no realiza reenvío de paquetes por su mismo, sino que mantiene la tabla de enrutamiento en el sistema operativo subyacente que, asumimos, será la que realice el reenvío de paquetes.

3.2.1.12. Información de topología redundante.

Para proveer de redundancia a la base de información de topología, el conjunto de enlaces anunciados de un nodo debería contener los enlaces a los nodos vecinos que no están en el conjunto de selectores MPR del nodo. El conjunto de enlaces anunciados debería contener enlaces a todos los vecinos del nodo. El conjunto mínimo de enlaces que un nodo debe anunciar en su mensaje TC es el formado por los enlaces a sus selectores MPR. El conjunto de enlaces anunciados se puede cosntruir de acuerdo a la siguiente regla basada en un parámetro local denominado TC_REDUNDANCY.

3.2.1.12.1. Parámetro TC_REDUNDANCY.

Este parámetro especifica, para el nodo local, la cantidad de información que se debe incluir en el mensaje TC. El parámetro debería ser interpretado de la siguiente manera:

a. Si el TC_REDUNDANCY de un nodo es 0, entonces el conjunto de enlaces anunciados del nodo es limitado por el conjunto de selectores MPR (ver sección 8.3).

b. Si el TC_REDUNDANCY es 1, el conjunto de enlaces anunciados es la unión del conjunto de MPR y del conjunto de selectores MPR.

c. Si el valor del parámetro es 2, entonces el conjunto de enlaces asociados, es el conjunto de vecinos completo.

Un nodo con “willingness” igual al WILL_NEVER_SHOULD tiene TC_REDUNDANCE igual a 0.

3.2.1.13. Redundancia MPR.

La Redundancia MPR especifica la capacidad de un nodo para seleccionar MPRS redundantes. La sección 4.5 especifica que un nodo debería hacer que su conjunto de MPR fuera lo más pequeño posible, para reducir la sobrecarga del protocolo. El criterio para seleccionar MPRS es, que todos los nodos estrictamente a 2 saltos deban ser accesibles por, al menos, un nodo MPR. La redundancia del conjunto de MPR afecta a la sobrecarga por: la cantidad de enlaces que pudieran estár siendo anunciados, la cantidad de nodos anunciando enlaces y la eficacia del mecanismo de envío de paquetes de los MPR. Por otra parte, la redundancia en el conjunto de MPR asegura que la accesibilidad para un nodo es anunciada por más nodos, difundiendo así enlaces adicionales por la red.

Mientras que un conjunto mínimo de MPRs disminuye la sobrecarga de la red, hay ocasiones en que conviene que este conjunto no sea mínimo, para obtener otros beneficios a costa de una mayor sobrecarga

3.2.1.13.1. Parámetro MPR_COVERAGE.

La cobertura MPR es definida por un parámetro local, el MPR_COVERAGE, especificando por cuantos nodos MPR debería ser cubierto un nodo estrictamente a 2 saltos. MPR_COVERAGE=1 asegura que mantendremos una sobrecarga mínima en la red, y hace que la selección MPR funcione como descrito en la sección 8.3.1. MPR_COVERAGE=m asegura que, de ser posible, un nodo selecciona su MPR de tal forma que todos los nodos estrictamente a 2 saltos son accesibles por al menos m MPR nodos. MPR_COVERAGE puede asumir cualquier valor de número entero> 0.

3.2.1.13.2. Computación MPR

Usando la cobertura MPR, la selección de heurísticas MPR se extiende desde lo descrito en la sección 8.3 por una definición:

Nodos poco cubiertos:

Un nodo poco cubierto es aquel nodo N2 que es cubierto por menos de MPR_COVERAGE (el valor del parámetro MPR_COVERAGE) nodos en N.

La heurística propuesta para seleccionar MPRs será por tanto como sigue:

1 Empezar con un conjunto MPR formado por todos los miembros en N con “willingness” igual a WILL_ALLWAYS

2 Calcular D(y), donde y es un miembro de N, para cada nodo de N.

3 Seleccionar como MPRs aquellos nodos en N que cubren a los nodos poco cubiertos de N2. Los nodes se eliminan de N2 para el resto de la computación.

4 Mientras existan nodos en N” que no son cubiertos por al menos MPR_COVERAGE nodos en el conjunto MPR :

4.1 Para cada nodo en N, calcular su alcanzabilidad, es decir, el número de nodos en N2 que no han sido cubiertos por al menos MPR_COVERAGE nodos en el conjunto MPR, y que sean alcanzables desde algún vecino a un salto.
4.2 Seleccionar como nodo MPR a aquel con el “willingness” más algo de entre los nodos en N con alcanzabilidad distinta de 0. En caso de multiples posibilidades, seleccionar el nodo que tenga la alcazabilidad más alta de todos los nodos de N2. En caso de haya múltiples nodos con la misma alcazabilidad, seleccionar como MPR a aquel nodo cuyo D(y) sea mayor. Eliminar los nodos

de N2 los nodos que ahora son cubiertos por MPR_COVERAGE nodos del conjunto MPR.

5. El conjunto MPR de un nodo se genera de la unión de los conjuntos MPR de cada interfaz. Como optimización, se puede procesar cada nodo, Y , del conjunto MPR en función de su N_willingness, empezando por el que tenga un N_willingness menor. Si todos los nodos en N2 siguen siendo cubiertos por al menos MPR_COVERAGE nodos en el conjunto de MPR excluyendo el nodo Y, y si N_willingness del nodo es menor que WILL_ALLWAYS, el nodo Y puede ser eliminado del conjunto MPR.

3.2.1.14. Número de Secuencia:

Los números de secuencia son usados para descartar paquetes antiguos, es decir, aquellos paquetes que son recibidos fuera de orden. Para ello se reservan en el formato del paquete unos cuantos bits que implementan un contador saturado, es decir, cuanto sobrepasan el máximo valor representable se pone a cero. Cada número representa el orden de los paquetes.

Esto supone un problema, ya que es frecuente que un paquete sobrepase el máximo. Un nodo que recibe ese paquete puede pensar que se trata un paquete más antiguo y por tanto lo rechace. Es importante que esto no suceda, para ello se introduce el concepto de MAXVALUE, un campo que indíca el tamaño máximo de paquetes de la secuencia. Este valor se tendrá en cuenta para saber que un paquete S1 es mayor que S2 si:

S1 > S2 AND S1 - S2 <= MAXVALUE/2

OR

S2 > S1 AND S2 - S1 > MAXVALUE/2

3.2.1.15. Seguridad:

- **Confidencialidad:** Ya hemos dicho varias veces que al ser un protocolo proactivo es necesario mantener un intercambio de paquetes de control para que cada nodo mantenga información parcial de la red. Es peligroso, en algunos casos, que un nodo no perteneciente a la red obtenga uno de estos paquetes, ya que obtendría información de la distribución de los nodos y posibilidad de reducir la integridad de la misma. Así que para algunas redes es deseable que se incluya una encriptación de los mensajes para evitar intercepciones fructuosas de los paquetes.

- **Integridad:** También es importante controlar los fallos de los nodos o la maldad de los mismos, ya que se pueden dar una serie de casos que, en caso de no estar controlados, afectaría de sobremanera a la integridad del protocolo, es decir, podría dejar de funcionar en determinadas situaciones. Por ejemplo,

- Nodos que manden mensajes HELLO o TC haciendose pasar por otros nodos de la red.
- Nodos que manden mensajes HELLO o TC con información equivocada.
- Un nodo que retransmite información de control errónea.
- Un nodo no selecciona los MPR de forma adecuada.
- Un nodo que si retransmite la información de control de forma correcta no lo hace con los paquetes de datos.
- Un nodo repite el envío de un mismo paquete de control constantemente.

La mayoría de estos casos se resolverán mediante el empleo de autenticación a la hora de mandar o recibir un mensaje, y en el último caso se empleará el uso de información temporal.

Generalmente, en alguna implementación, se transmiten firmas digitales y algúnas otras formas de control de seguridad como mensajes separados de los del protocolo. De esta forma pueden coexistir nodos fiables y maliciosos en la misma red y aumentar la integridad de la misma de forma considerable. Pero, es inevitable mencionar que aumentarán las colisiones debido al aumento del tráfico entre los nodos.

También es muy importante recordar que un nodo emite paquetes de control a sus vecinos, es decir: point-to-multi-point. Para esto es vital un sistema de autenticación, ya que si no se podrían generar enormes problemas de información falsa con tan solo un envío múltiple de un paquete.

3.2.1.16. Control de tráfico y congestión:

Al ser un protocolo clasificado como proactivo el control de tráfico se hace de forma natural. Por una lado el empleo de nodos MPR evita la inundación de paquetes en la red, común en protocolos reactivos. Además, que la existencia de nodos MPR produce que el nº de paquetes para adquirir la información sea menor, al anunciarse solamente los nodos selectores de MPRs.

El envío de paquetes de control aumenta el tráfico, por tanto es muy importante ajustar el tiempo de reenvío de este tipo de paquetes. Un tiempo muy corto entre paquete y paquete producirá un tráfico muy denso y aumentará las colisiones entre paquetes. Por el contrario un tiempo de reenvío elevado producirá que los nodos, posiblemente, tengan información poco actual del estado de sus vecinos, haciendo inútil el concepto de proactividad.

Es posible, además de sencillo, implementar la posibilidad de que los nodos envíen un mensaje de desconexión de la red. De esta forma temporalmente se produce un aumento del tráfico pero ayudará a tener información más actualizada de la red.

3.2.1.17. OLSRD

Una implementación actual y con mucho seguimiento se encuentra en la página web www.olsr.org. En ella se nos explica que está implementación comenzó como la tesis de un estudiante, Andreas Tønnesen. Sus objetivos, principalmente, eran los de conseguir un código bien estructurado a la par que mantenible y portable.

Así, el demonio OLSR está disponible para varios sistemas operativos, como son: GNU/Linux, Windows, OS X, FreeBSD, OpenBSD y NetBSD. Todo esto es una pequeña prueba del seguimiento que está recibiendo esta implementación. Aparte de que se dispone de la posibilidad de cargar *plugins* sobre el demonio para aumentar o modificar las funcionalidades que ofrece. Por ejemplo, se puede poner un *plugin* para cambiar el formato de los paquetes o bien cambiar la forma de inundación de los mensajes para seleccionar los MPRs

Comentaremos algunos detalles, pero antes recordamos que toda la información de este documento se ha obtenido del documento RFC3626. En el no solo se explica con detalle el funcionamiento del protocolo sino también el de esta implementación. El protocolo, debido a la información contenida en cada nodo es capaz de mandar un paquete de un nodo a otro por el camino más corto. Ahora bien, puede darse el caso de que existan dos caminos alternativos desde el mismo origen al mismo destino, uno de ellos pasando por un solo nodo defectuoso, y la otra alternativa pasando por dos nodos en excelentes condiciones. Con nodo defectuoso entendemos un nodo con una alta probabilidad de pérdidas de paquetes. Seguramente, a pesar de realizar dos saltos, el segundo camino sea más corto en tiempo. OLSRD sí realiza está distinción y está implementado de forma que distinga de los nodos problemáticos de los nodos en buenas condiciones.

Para ello definimos el concepto de *Link Quality* que mide el porcentaje de paquetes de control que un nodo ha perdido. Para ello, como sabemos, fijado en la configuración, que el tiempo de envío de un paquete Hello se realiza cada dos segundos, un nodo puede saber cuantos paquetes ha recibido y cuantos debería haber recibido:

$$LQ = N^{o} \text{ paquetes recibidos} / N^{o} \text{ paquetes Teoricos}$$

Aún así, también es importante tener en cuenta el porcentaje de paquetes enviados por un nodo que han llegado bien a su destino. Para ello, un nodo lleva la cuenta de cuantos paquetes de control ha enviado y cuantas confirmaciones ha recibido. Definimos este porcentaje como *Neighbor Link Quality:*

NLQ = Nª ACK Recibidos / Nº Paquetes Enviados.

Este último concepto es algo menos estricto que el anterior, ya que no recibir una confirmación de un paquete no significa haber perdido dicho paquete, si no que se podría haber perdido la propia confirmación. Por tanto, para definir la calidad de un nodo, OLSRD usa una aproximación en base a los dos conceptos anteriores:

Calidad = LQ x NLQ

Cuanto más cerca de uno (o 100) menor será la probabilidad de perdida de un paquete por parte de un nodo y, por tanto, más fiable será la retransmisión de paquetes a través de dicho nodo.

3.2.1.17.1. Valores y constantes propuestas:

Destacamos las más importantes:

- *HELLO_INTERVAL = 2 seconds*
- *REFRESH_INTERVAL = 2 seconds*
- *TC_INTERVAL = 5 seconds*
- *Constante C = 1/16 seconds.*

La constante C se usa entre otras cosas para determiner el tiempo de validez de la información de un nodo. Es decir, su *"validity time (vtime)"*. Recordamos que ese valor es un campo de las cabeceras de los mensajes Hello, y que indica el tiempo durante el cual la información que contiene el mensaje es válida, desde que se recibe dicho mensaje.

vTime = C(1+a/16)*2^b*

Donde "a" representa los 4 bits más significativos de ese campo, y "b" representa los 4 menos significativos.

3.2.1.17.2. Simulación:

OLSR-Switch es una aplicación de enrutamiento de tráfico para permitir a distintas instancias de OLSRD conectarse y comunicarse a través de una interfaz TCP. Permite asignar la calidad, comentada anteriormente, a cada nodo participante en la simulación. Así, se distinguirán 3 tipos de nodos:

- Nodo Abierto: Con calidad 100, todo enrutamiento escogerá, si es posible, este nodo antes que cualquier otro.

- Nodo Cerrado: Con calidad 0, al revés que el caso anterior. Jamás se escogerá este nodo en el reenvío de información.

- Nodo Variable: Con calidad entre 1-99. Por ejemplo, un valor de 25 significaría que este nodo pierde en torno al 75% de los paquetes que reenvía.

3.2.1.18. Conclusiones:

Cabe destacar que estamos ante uno de los protocolos más importantes y con más repercusión en este tipo de redes que estudiamos. Simplemente con leer el apartado anterior, donde hablamos de una conocida implementación, observamos que se trata de uno de los protocolos con mejores resultados.

El hecho de que sea proactivo, produce que el tiempo de retardo en el envío de paquetes sea bastante pequeño. Recordemos, que una vez escogidos el conjunto de MPRs un paquete viajará de un nodo MPR a otro nodo MPR hasta llegar a su destino y lo hará (salvo posibles modificaciones en la implementación) por el camino más corto posible.

Durante el diseño de un protocolo se establecen una serie de objetivos que han de tratar de cumplirse lo mejor posible. El grado de cumplimiento de estos aspectos es el que determina que un protocolo pueda considerarse mejor o peor que otro.

En este protocolo, a pesar de la distinción de los MPRs de los nodos normales, ningún nodo tiene prioridad respecto a otros nodos a la hora de enviar un mensaje a otro nodo. Es decir, es un protocolo *justo*. Destacamos que los MPR son simplemente, por así decirlo, retransmisores de la información, unas veces de un punto a otro, y otras de un punto a varios puntos. Pero, en ningún momento gozarán de otra prioridad para transmitir.

También es un protocolo *estable*, ya que es posible que se caiga algún nodo de la red y esta no se bloqueará ni perderá información. Tan solo requerirá una actualización de la información de la red en cada nodo, pero el protocolo seguirá funcionando correctamente. Está actualización, como decimos, se hace de forma correcta gracias al reenvío crónico de los paquetes de control. Volvemos a destacar que aunque sea un nodo MPR el que se desconecte de la red, ésta seguirá siendo estable.

Dada la naturaleza proactiva del protocolo es inevitable que multitud de paquetes de control circulen por la red, pero, al contrario de lo que se puede pensar, el protocolo puede seguir funcionando perfectamente con una perdida aceptable (esto es, no desmesurada) de paquetes de control. Estamos, por tanto, ante un protocolo *robusto*.

La *Calidad del Servicio* (*QoS*) ofrecida por este protocolo también es bastante elevada. De hecho, al compararlo con algún protocolo reactivo se aprecian grandes diferencias, principalmente debidas el tiempo de entrega de los paquetes, ya que, salvo en algunos casos y gracias a la cache de rutas de los nodos, generalmente el tiempo de búsqueda de la ruta producirá grandes

retrasos. No así en una red proactiva, un nodo recibe un paquete y la reenvía instantáneamente empleando la información local contenida en si mismo.

Más complicado se pone el tema al hablar sobre la ***productividad*** de la misma. Hemos comentado en varias ocasiones que generar muchos paquetes de control (intervalo pequeño de tiempo entre el envío de paquetes sucesivos) puede incrementar de sobremanera el tráfico, decrementando la productividad. Esto se puede solucionar aumentando dicho intervalo de tiempo, lo que puede provocar que la información temporal sobre la topología de la red contenida en cada nodo sea antigua y por tanto el protocolo dejaría de ser adaptativo a los cambios en la topología de la red. Por tanto es imprescindible optimizar este intervalo de tiempo. De hecho, sería óptimo que fuera ajustable automáticamente de acuerdo a las características momentáneas de dicha red. A pesar de todo, la productividad de este protocolo es bastante aceptable. En ningún caso estamos ante un óbice para su uso e implementación.

Finalmente nos queda comentar el aspecto que trata el ***consumo de energía.*** Quizá sea el aspecto más embarazoso (y en muchas ocasiones el más ignorado) a tratar. El envío constante de mensajes de control aumenta bastante el consumo de energía. Además, los MPR procesan una cantidad mucho más elevada de mensajes, lo que produce que su gasto de batería sea mucho mayor.

Hemos relacionado una serie de aspectos concretos con el protocolo, pero queda la relación más importante que se puede realizar con las MANET, y es comentar como afecta la ***movilidad*** de los nodos al estado de la red. Realmente, no hay ningún problema con este aspecto, ya que cada nodo se dará cuenta rápidamente de los cambios. La movilidad de los nodos afectará levemente el rendimiento (actualización de las tablas de información de los nodos y vecinos) pero no afectará ni a la estabilidad ni robustez.

Es también importante recordar la seguridad. Para redes importantes que requieran seguridad y control se incluirán modos de autenticación.

Al igual que hemos comenzado este apartado lo vamos a finalizar, mencionando que, seguramente, pocos protocolos para redes inalámbricas hayan despertado tanto interés y tenido tanta repercusión, pero, no por ello podemos afirmar que es el mejor protocolo disponible actualmente.

3.3. Protocolos Reactivos:

3.3.1. DSR: Dynamic Source Routing Protocol

3.3.1.1. Introducción:

David B. Jhonson ideó en 1994 el DSR: Dynamic Source Protocol. Su objetivo era conseguir un protocolo sencillo y eficiente, que tratase de aprovechar al máximo las características de las MANETs.

Es un protocolo reactivo, bajo demanda, es decir no guarda información sobre el estado de la red, salvo rutas ya calculadas en “caches” destinados a ellas, es decir, la topología de la red varia sin que los nodos sepan la nueva configuración, pero, en el momento de una transmisión de un mensaje el nodo iniciará una serie de operaciones para averiguar la ruta hasta su destino. Como decimos, cada nodo dispondrá de una cache de rutas en las que puede almacenar distintas rutas, bien averiguadas o bien conocidas de alguna forma que comentamos a continuación.

DSR está compuesto por 2 mecanismos, uno es el Descubrimiento de la Ruta (Route Discovery) y el otro es el Mantenimiento de la Ruta (Route Maintenance) Los explicaremos con más detalle, pero adelantamos que el primero se encarga de descubrir la ruta por la que viajarán los paquetes de un nodo origen a un nodo destino. El segundo verificará que la ruta escogida siegue siendo factible y que los paquetes están llegando al destino deseado de forma correcta.

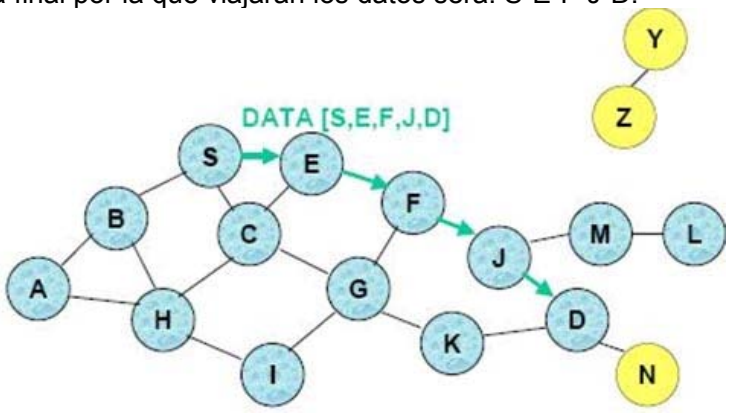
3.3.1.2. Route Discovery:

Se produce cuando un nodo S quiere enviar paquetes de datos a un destino D, pero no conoce la ruta hacia D, entonces, S inicia este mecanismo. El nodo origen S empieza a mandar paquetes a sus vecinos, estos paquetes son denominados Route Request. Este paquete RREQ contiene el identificador del nodo origen y destino, e incluye también la ruta parcialmente calculada.

Cada nodo que recibe un RREQ va a añadir en la ruta contenida en el paquete su identificador y mandará a sus vecinos el nuevo RREQ. Este proceso se itera por los distintos nodos hasta que uno de estos paquetes lo reciba el destino. Este, al recibir el RREQ comenzará un sistema análogo mandando un Route Reply (RREP). Este proceso consiste en invertir la ruta calculada para hacer llegar al origen la ruta por la que viajarán los datos. En este proceso hay que tener en cuenta la direccionalidad de los nodos, ya que un nodo puede ser unidireccional o bidireccional. En el primer caso, un nodo A puede comunicarse y enviar datos a un nodo B pero este no puede comunicarse con A. En el segundo caso la comunicación se puede hacer en ambos sentidos. Hay que tener en cuenta esto porque al realizar el Route Reply se invierte la ruta calculada, por tanto, si uno de los nodos es unidireccional implica que no puede mandar el RREP al nodo correspondiente, y se ve obligado a iniciar otro Route Request para conseguir que el RREP llegue al nodo deseado y así proseguir con el Route Reply.

Recapitulando, uno nodo A quiere transmitir un mensaje a un nodo B. Primero se inunda la red con un Route Request, un mensaje que irá circulando por los distintos nodos de la red almacenando la ruta calculada. Cuando uno de estos paquetes llega al nodo B este responde al Route Request con un Route Reply, es decir, manda un mensaje que tiene que llegar al nodo A para informarle de la ruta por la que tienen que viajar los paquetes. El nodo A, al recibir el RREP introduce en su cache la ruta incluida en el propio RREP. Cuando A envía un mensaje de datos a D, toda la ruta que ha de seguir se incluye en la cabecera, de ahí el nombre de Source Routing. Los nodos intermedios usan la ruta incluida en la cabecera del mensaje para saber a que nodo mandar el mensaje.

En la siguiente figura vemos una posible configuración de los nodos de una red. De color azul vemos todos aquellos que reciben el Route Request, pero la ruta final por la que viajarán los datos será: S-E-F-J-D.



3.3.1.2.1. Caching Overhead:

Hemos comentado que cada nodo va a disponer de una cache donde almacenarán las distintas rutas que conocen. Es importante este concepto ya que el conocer la ruta de antemano puede reducir el tiempo de entrega de los paquetes de sobremanera. Por tanto, tenemos que saber que rutas incluímos en la cache:

- **Paquetes "escuchados"**: Un nodo A puede estar escuchando la retransmisión de un paquete de un nodo S a un nodo D sin que A intervenga en la retransimisión.

- **Ruta de envío en el paquete "fowarded"**: Un nodo S está retransmitiendo un paquete a un nodo D usando otro nodo A entre medias, este nodo A en el momento de la retransmisión del paquete puede almacenar la ruta de esa retransmisión.

- **Ruta acumulada durante el Route Request:** Parecido al anterior solo que en este caso el nodo intermedio no retransmite un mensaje si no que ha recibido un Route Request con una ruta parcial y, aparte de

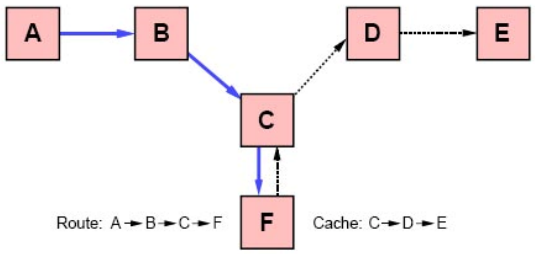
proseguir el RREQ almacena la ruta que iba en la cabecera del propio RREQ.

- **Ruta acumulada en el Route Reply:** Análogo al anterior pero durante la propagación del RREP.

3.3.1.2.2. RREQ con Cache:

Para ganar tiempo en las transmisiones ya hemos comentado lo importante que es el buen uso de la cache, en este apartado vamos a destacar otro aspecto muy importante del uso de la cache. Cuando se inicia un Route Discovery y un nodo recibe el RREQ mira si tiene en la cache el nodo destino, lógicamente, pueden darse dos casos, que el nodos se encuentre o, al contrario, que no se encuentre. Si no se encuentra se prosigue con el Route Request de forma normal, es decir, el nodo seguirá inundando la red con RREQs.

Por el contrario, si el nodo destino se encuentra en la cache del nodo *relay* (nodo intermedio) este nodo puede unir la ruta almacenada hasta el momento en el RREQ con la almacenada en la cache y cancelar el Route Request. De esta forma comenzaría directamente el Route Reply por parte del nodo *relay*. Es importante que se compruebe que no se repite ningún nodo en la nueva ruta generada. Por ejemplo, en la siguiente figura, el nodo A inicia un Route Request por que quiere mandar un paquete al nodo E. Suponemos que en la cache del nodo F tiene ya la ruta calculada desde F hasta E. En un determinado instante de tiempo F recibirá un Route Request con la ruta calculada hasta el momento, es decir, A-B-C-F. F comprueba su cache y como ya tiene la ruta calculada hasta el nodo E simplemente une ambas rutas, generando la siguiente ruta: A-B-C-F-C-D-E, como el nodo C está repetido en la secuencia (y produce un bucle bastante inútil) F antes de realizar el Route Reply, retoca la ruta y genera la A-B-C-D-E.

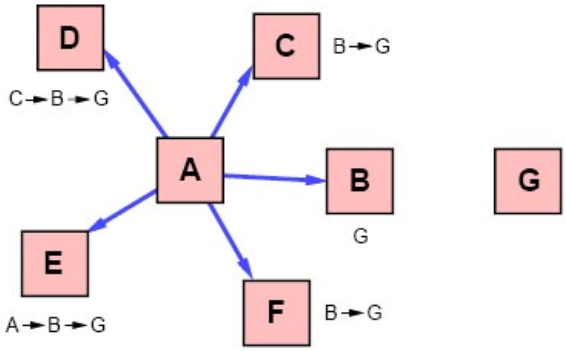


3.3.1.2.3. Route reply Storms:

No es extraño, ni mucho menos, considerar el caso donde varios nodos reciban un Route Request y varios de ellos tengan en su cache la ruta resultante. Comenzando por tanto, al mismo tiempo, varias Route Reply. Esto produce que varias transmisiones con el RREP sucedan al unísono

malgastando el ancho de banda y aumentando las colisiones entre los paquetes.

Una posible solución, bastante sencilla, es que cada nodo espere un tiempo aleatorio en función del número de saltos hasta el destino. El nodo irá recogiendo poco a poco las rutas, de forma más ordenada y reduciendo las colisiones. Además, rechazará aquellas rutas más largas.



3.3.1.2.4. Route Request: Hop Limits:

En el formato del paquete del Route Request se incluye un campo que indica el máximo número de saltos de un paquete RREQ. Este campo se decrementará en cada salto que se produzca durante el Route Request, y si llega a cero se deshecha el paquete.

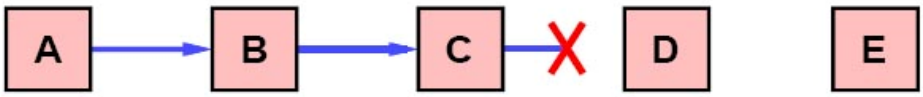
Otra posibilidad es la emisión del RREQ de forma "*Anillo Expansible*". Es decir el emisor inicia el Route Request con este campo igual a uno, si no recibe respuesta, dobla el campo y comienza de nuevo otro Route Request. Este proceso se itera sucesivamente hasta que, por fin, el nodo origen reciba el Route Reply.

Tras unas cuantas horas leyendo y releyendo información sobre este protocolo podemos empezar a hacer unas cuantas consideraciones acerca de su posible implementación en el proyecto, teniendo en cuenta ventajas y desventajas. Vamos a hacer un análisis algo más detallado.

3.3.1.3. Route Manteinance:

Una vez finalizada la etapa anterior, es decir, el paquete de Route Reply ha llegado al origen con la ruta resultante, el origen comienza el envío de los datos por la ruta. Es en este momento cuando comienza la etapa de mantenimiento. Esta etapa comprueba constantemente que los paquetes llegan al destino deseado. Para explicarlo de forma más clara, vamos a observar la siguiente figura. En ese caso A está mandando paquetes a E. En cada tramo del envío un nodo se hace cargo de comprobar que el paquete que envía es

recibido por el siguiente nodo. Es decir, A comprueba que B ha recibido el paquete, B comprueba que C lo recibe, etc.



Una posible forma de comprobar esto es "escuchando" las retransmisiones, es decir, para que A sepa si B ha recibido el paquete, A puede escuchar que B lo está retransmitiendo hacia C. Aún así se suele incluir un bit de comprobación en el paquete, para que el siguiente nodo responda un ACK para confirmar que ha recibido correctamente el paquete. En este caso hay que tener en cuenta la simetría de la conexión, es decir, controlar si el enlace es unidireccional o bidireccional. En caso de ser bidireccional no existiría ningún problema, pero, si fuera unidireccional el nodo que tiene que mandar la comprobación tiene que buscar rutas alternativas. Esto, como veremos, puede traer grandes inconvenientes.

Cualquier nodo implicado en la retransmisión de paquetes tiene que tener un sistema de control de envíos, se implementa de forma sencilla controlando el número máximo de envíos por paquete. En caso de que un nodo alcance esta cifra considerará que el siguiente nodo está fallando, y por tanto mandará un Route Error hacia el nodo origen. Los nodos actualizarán la cache para cambiar las rutas y que eviten ese enlace (ahora roto) El nodo origen tiene que reenviar la cache, tiene dos posibilidades, puede comprobar la cache para ver si tiene una ruta alternativa, al realizar el Route Request es muy probable que recibiera varios Route Replys alternativos. Si en la cache no tiene ninguna alternativa posible el nodo origen se verá obligado a iniciar otro Route Request y encontrar otra ruta.

3.3.1.3.1. Salvación de paquetes:

Cuando un nodo falla en el envío hemos comentado una solución que consiste en avisar al nodo origen y que comience otro reenvío. Ahora bien, una posibilidad es que el nodo que falla en la entrega mande el Route Error hacia el origen, pero intente "salvar" el paquete, es decir, sería el propio nodo intermedio el que buscase una ruta alternativa. Así, cambiaría la ruta original del paquete por la nueva ruta escogida de la cache. Posteriormente marcaría el paquete como "*salvado*". Este marcado se realiza por si otro nodo posterior al que lo ha marcado fallase en la entrega y no existiera el marcado, podrían producirse bucles, y que, por tanto, el paquete nunca llegase a su destino.

Otra alternativa es dejar el prefijo de la ruta hasta el error y añadir un sufijo hasta el destino, así se evitan los bucles y el backtracking. Además, en este caso no se necesita el marcado del paquete.

3.3.1.3.2. Acortamiento de Rutas:

Como hemos dicho al principio uno de nuestros principales objetivos es que sea un protocolo eficiente. Hasta el momento hemos comentado siempre la importancia de la optimización de la cache, además de esta forma existe un factor bastante obvio que reduce el tiempo de envío de los paquetes y es, obviamente, reducir el número de retransmisiones de un paquete.

Para ello, si durante el envío de los paquetes un nodo detecta una ruta más corta ese nodo actualizará la ruta para reducir el número de retransimisiones. Por ejemplo, en el caso de la figura el Route Request ha determinado que la ruta para entregar un paquete de A a D es A-B-C-D pero también se da el caso que A puede comunicarse directamente con C, por tanto, la ruta más corta es A-C-D. Este proceso automático de reducción de rutas se produce durante el Route Reply.

3.3.1.3.3. Propagación incremental de enlaces rotos:

En caso de que se haya descubierto un error durante una transmisión de un paquete se propaga hacia el origen un Route Error. Una posible implementación es que el origen reenvíe el mismo route Error a sus vecinos en el siguiente Route Request que realice. De esta forma todas las rutas que se generen no contendrán en ningún caso el enlace roto.

3.3.1.3.4. Información negativa en la cache:

Para reducir el número de errores se puede almacenar en la cache información sobre enlaces rotos, nodos problemáticos. Se podría simplemente eliminar dicha entrada de la cache, pero de esta forma se garantiza que un Route Reply no contendrá en ningún caso información que ya ha generado errores (por tanto, con elevada probabilidad de volver a darlos).

3.3.1.4. Multicasting

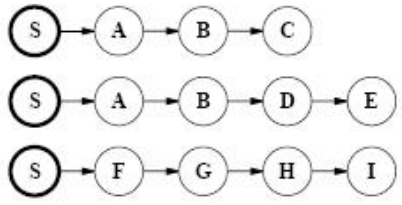
Este modo de envío de paquetes no ha sido considerado en ninguna implementación del protocolo para ser resuelto de una manera eficiente. Las soluciones propuestas son soluciones sin control de información como pueden ser aquellas con uso de árboles o grafos multicast. Una manera poco efectiva de hacerlo es mediante un flooding con control de saltos y filtrado de paquetes. Consiste en limitar el número de saltos de un paquete por la red (véase el Hop Limits, explicado anteriormente) El filtrado, en cambio, consiste en que un nodo perteneciente a un conjunto multicast aceptará aquellos paquetes que vayan destinados a el. Pero, aquellos que no pertenezcan a ese conjunto los recibirán igualmente pero, al no pertencer, harán caso omiso del paquete. Por eso no es eficiente, porque el paquete se manda a todos los nodos posibles sin distinguir durante los envíos si se envían a un destino adecuado o no.

3.3.1.5. Conclusiones:

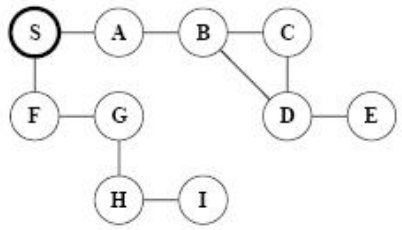
Cómo ya hemos repetido, es un protocolo reactivo, bajo demanda, es decir, no guarda información sobre la red (salvo alguna ruta en una "cache" destinada a ello) si no que cuando desea enviar algún paquete procede a buscar la ruta hasta su destino. Esto es un arma de doble filo, como sabemos, la principal caracteristica de una MANET es que sus nodos son móviles, así que el DSR al no guardar información de la red no requiere que se manden contínuamente paquetes sobre el estado de la red. Ya que solo requerirá dicha información en el momento del envío (y ni si quiera información completa, si no tan solo la estrictamente necesaria para unir un origen con uno destino) A priori, esto suena muy interesante, ya que reduce la sobrecarga de la red y por tanto reduce el número de colisiones en la red (menos paquetes de control, menos mensajes circulando en un instante determinado, ergo, menos colisiones) Sin embargo, a posteriori la idea deja de ser tan atractiva, ya que, desde nuestro punto de vista, el número de inconvenientes que tiene conlleva la implementación sugerida es bastante mayor que las ventajas que trae. A saber, el tiempo de obtención de una ruta de datos puede ser vital, ya que en determinados casos (para comunicaciones en tiempo real principalmente) el "delay" de entrega de un paquete ha de ser mínimo.

Hemos comentado que la implementación admite tanto nodos direccionales como nodos bi-direccionales, esto de nuevo no es tan bueno como cabe esperar, ya que puede producir que un nodo al enviar un Route Reply no pueda comunicarse *directamente* con el nodo que tiene que recibir ese Route Reply, en cuyo caso debería de iniciar otra búsqueda de la ruta. Esto, indudablemente, aumenta de sobremanera el tiempo de entrega de un paquete.

Cabe destacar la importancia en este protocolo del papel tan importante que juega la cache de rutas de cada nodo. Hemos dicho que el tiempo de entrega de un paquete siempre debe de ser lo más pequeño posible, una cache correctamente configurada puede suponer grandes ganancias de tiempo. No almacenar nodos rotos pero sí almacenar el máximo posible número de rutas, tanto escuchadas, como efectuadas o simplemente direcciones que un nodo capta por hacer de "relay" en la retransmisión de distintos paquetes. Ahora bien, no podemos olvidar que en la MANET no solo son móviles los nodos, si no que, generalmente, también son de escasa memoria. Almacenar muchas direcciones implica un mayor tiempo de acceso a la cache (cuanto más grande sea más complicado y más retardo se obtiene en el acceso) y a la vez mayor es el espacio en memoria requerido.



(a) Path Cache



(b) Link Cache

En definitiva, es un protocolo sencillo y amigable, pero no del todo adecuado para redes grandes con alta movilidad, donde el rendimiento de la red se ve claramente perjudicado.

3.3.2. DYMO (Dynamic MANET On-Demand)

3.3.2.1. Introducción.

El protocolo DYMO (*Dynamic MANET On-demand*) es un protocolo reactivo que está en fase de desarrollo, por lo que actualmente no está estandarizado, pero está previsto que así lo sea en un futuro, tiene categoría de Standards Track. Hereda características de sus antecesores AODV y DSR, al ser un protocolo reactivo introduce una latencia al descubrir una ruta, pero lo contrarresta con el poco tráfico de control utilizado en la red. En este documento se explica el funcionamiento del protocolo y sus implementaciones. Se considera una evolución del AODV.

Es un protocolo joven que no está del todo desarrollado. La última versión del draft es la número 7, y está fechada en el 9 de febrero de 2007. Esta especificación está desarrollada gracias a Ian Chakeres de Boeing, Elizabeth M. Belding-Royer de la Universidad de Santa Barbara y Charles Perkins del grupo de investigación de Nokia.

El protocolo DYMO es un protocolo reactivo que no envía paquetes de control si no está realizando funciones de encaminamiento, transmisión o recepción de información, lo que es bueno para redes en la que los recursos son escasos. Dicho protocolo es compatible con las versiones 4 y 6 del protocolo IP. Actualmente se está desarrollando una versión reducida para redes de sensores, llamada DYMO-LOW, donde el objetivo principal a tener en cuenta es el bajo consumo de energía y la reducida capacidad de proceso que tienen estos sensores.

Las operaciones básicas del protocolo DYMO son el descubrimiento de ruta y el mantenimiento de ruta. Durante el descubrimiento de ruta el nodo origen inicia la diseminación de un Route Request (RREQ) en todas partes de la red para encontrar el nodo objetivo. Durante este proceso de diseminación, cada nodo intermedio registra una ruta al nodo del que procede. Cuando el nodo objetivo recibe el RREQ, este responde con un Route Reply (RREP) enviado al nodo origen. Cada nodo que recibe el RREP registra una ruta al nodo objetivo, y luego el RREP es unicast hacia el nodo origen. Cuando el nodo origen recibe el RREP, las rutas entonces han sido establecidas entre el nodo origen y el nodo objetivo en ambas direcciones.

Para reaccionar a cambios de topología de red los nodos mantienen sus rutas y supervisan los enlaces por los cuales circula el tráfico de datos. Cuando un paquete de datos es recibido, para notificar que una ruta se ha perdido, se envía un Error de Ruta (RERR) a la fuente de paquete para indicar que la ruta actual se ha roto. Cuando la fuente recibe el RERR, sabe que debe realizar un Route Discovery si todavía tiene paquetes para entregar. El UERR (*Unsupportedelement Error*) es un mensaje de error no soportado por el

protocolo. A éste último no le ha encontrado utilidad, por lo que en una de las últimas reuniones del grupo de trabajo de MANET se ha decido suprimirlo del draft (*dicho cambio se vió reflejado en la versión del draft número 4*). Dependiendo de la técnica de mantenimiento de las rutas empleada, también pueden existir mensajes de hellos o de reconocimiento, todos estos paquetes son enviados en UDP por un puerto To Be Determined (TBD) - El NIST-DYMO utiliza el puerto UDP número 462, mientras el DYMOUM el puerto UDP número 653.

DYMO usa números de secuencia para asegurar la ausencia de ciclos en las rutas. Los números de secuencia permiten a los nodos determinar el orden (el pedido) de mensajes de descubrimiento de ruta DYMO, así evitando el empleo de información de encaminamiento obsoleta.

3.3.2.2. Aplicabilidad.

El protocolo DYMO está especialmente diseñado para redes ad hoc con movilidad variable. DYMO gestiona un amplio rango de patrones de movilidad para determinar dinámicamente rutas bajo demanda. Igualmente, tiene capacidad para manejar diferentes patrones de tráfico. En redes grandes tiene un mejor comportamiento en escenarios donde cada nodo sólo se comunica con una limitada porción de los demás.

DYMO es aplicable a sistemas con limitaciones de memoria, gracias a que solo es necesario mantener un pequeño conjunto de rutas. Concretamente, sólo se mantiene información de enrutamiento relacionada con fuentes y destinos activos, a diferencia de otros protocolos que mantienen información de todos los nodos que forman parte del sistema.

El algoritmo de encaminamiento en DYMO puede ser realizado en otras capas además de la de red, empleando las direcciones de capa oportunas. Hay que tener especial cuidado con no cambiar el formato del paquete, puesto que si se le aplica un formato diferente de los que se explicarán más adelante (los propios del protocolo), el resto de los nodos no podrán procesar dicho paquete.

3.3.2.3. Terminología.

Vamos a emplear los mismos términos que se emplean en los diferentes drafts (documentos de divulgación publicados en Internet con un período de validez máximo de seis meses) que han sido publicados para este protocolo. Como aclaración, en estos documentos las palabras clave "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", y "OPTIONAL" se emplean con el significado que se les aplica en el RFC 2119 (“Key words for use in RFCs to Indicate Requirement Levels” de S. Bradner, 1997).

A continuación definimos el significado de algunos términos que se emplearán de aquí en adelante:

- *DYMO Sequence Number (SeqNum):* Cada nodo mantiene un número de secuencia DYMO. Éste es utilizado por los otros nodos para identificar el orden de la información de encaminamiento generada por un nodo y garantizar la ausencia de ciclos.

- *Forwarding Route* Una ruta que se usa para enviar paquetes de datos. Estas rutas generalmente se mantienen en una base de información de envío de datos (FIB, *forwarding information base*) o en la tabla de encaminamiento del kernel.

- *Hop Count (HopCnt)* El número de saltos que un mensaje o fragmento de información ha tomado.

- *Originating Node (OrigNode)* Es el nodo que crea el mensaje DYMO para transmitir cierta información.

- *Route Error (RERR)* Un nodo genera y propaga un RERR para indicar que no puede realizar un envío a una o varias direcciones.

- *Route Reply (RREP)* UN RREP se emplea para propagar información de encaminamiento desde el OrigNode del RREP al TargetNode y los nodos que se encuentran entre ellos.

- *Route Request (RREQ)* Un nodo (el OrigNode del RREQ) genera un RREQ para descubrir una ruta válida a una dirección de destino concreta, el RREQ TargetNode. Cuando un nodo procesa un RREQ, aprende información de encaminamiento sobre como alcanzar el OrigNode.

- *Target Node (TargetNode)* Es el destino final del mensaje.

- *This Node (ThisNode)* ThisNode corresponde al nodo que actualmente está procesando o realizando cálculos sobre un mensaje.

- Type-Length-Value structure (TLV) Una forma genérica de representar información.

- Unreachable Node (UnreachableNode) Un nodo para el cual ThisNode no tiene una ruta de envío.

3.3.2.4. Descripción.

3.3.2.4.1. Estructuras de datos.

3.3.2.4.1.1. Entrada de la tabla de encaminamiento.

Una entrada de la tabla de encaminamiento tiene los siguientes campos:

- Route.Address: la dirección IP de destino del nodo asociado con la entrada de la tabla de encaminamiento.

- Route.SeqNum: el SeqNum asociado con esta información de enrutamiento.

- Route.NextHopAddress: la dirección IP del siguiente nodo en el camino a la Route.Adress.

- Route.NextHopInterface: el interfaz usado para enviar paquetes a la Route.Adress.

- Route.Broken: un flag que indica si la ruta está rota. Este flag se activa si no se puede realizar el siguiente salto o al procesar un RERR.

Los siguientes campos son opcionales:

- Route.HopCnt: el número de saltos entre nodos intermedios que se han tomado en dirección al nodo destino. Este valor puede ayudar a determinar si una información de enrutamiento es mejor que la que ya se conoce.

- Route.Prefix: indica que la dirección asociada es una dirección de red, más que una dirección de un host. El valor es la longitud de la máscara de red. Si un bloque de dirección no tiene asociado un PREFIX_LENGTH TLV, el prefijo debe tomar la misma longitud que la dirección (en bits).

3.3.2.4.1.2. Mensajes DYMO.

Notación empleada para referirse a cada campo.

Information Location	Notational Prefix
IP header	IP.
UDP header	UDP.
packetbb message header	MsgHdr.
packetbb message TLV	MsgTLV.
packetbb address blocks	AddBlk.
packetbb address block TLV	AddTLV.

3.3.2.4.1.2.1. Estructura general de los paquetes y mensajes MANET.

Un mensaje está compuesto por un message header, message TLV block, y cero o más address blocks. Cada uno de estos adress block tiene asociado un TLV block.

Todos los mensajes DYMO especificados en este documento son enviados usando UDP al puerto de destino TBD.

La mayoría de los mensajes DYMO son enviados con el link a dirección local multicast LL_ALL_MANET_ROUTER como valor de la dirección IP. A los mensajes Unicast se les asigna la RouteNextHopAdress de la ruta al TargetNode como dirección IP.

La longitud de una dirección IP (32 bits para IPv4 y 128 bits para IPv6) dentro de un mensaje DYMO depende de la cabecera del paquete IP que contiene el mensaje DYMO. Por ejemplo, si la cabecera IP usa direcciones IPv6, todos los mensajes y direcciones contenidos en el segmento de datos usan direcciones IPv6. En caso de mezclar direcciones IPv4 e IPv6, las direcciones IPv4 se incluyen en direcciones IPv6.

3.3.2.4.1.2.2. Routing Messages (RM) - RREQ & RREP

Los Routing Message se usan para diseminar información de encaminamento. Hay dos tipo de mensajes DYMO que se consideran Routing Messages: RREQ y RREP. Contienen información muy similar, pero se procesan de manera diferente. La principal diferencia entre los dos es que los mensajes RREQ solicitan un RREP, mientras que RREP es la respuesta a RREQ.

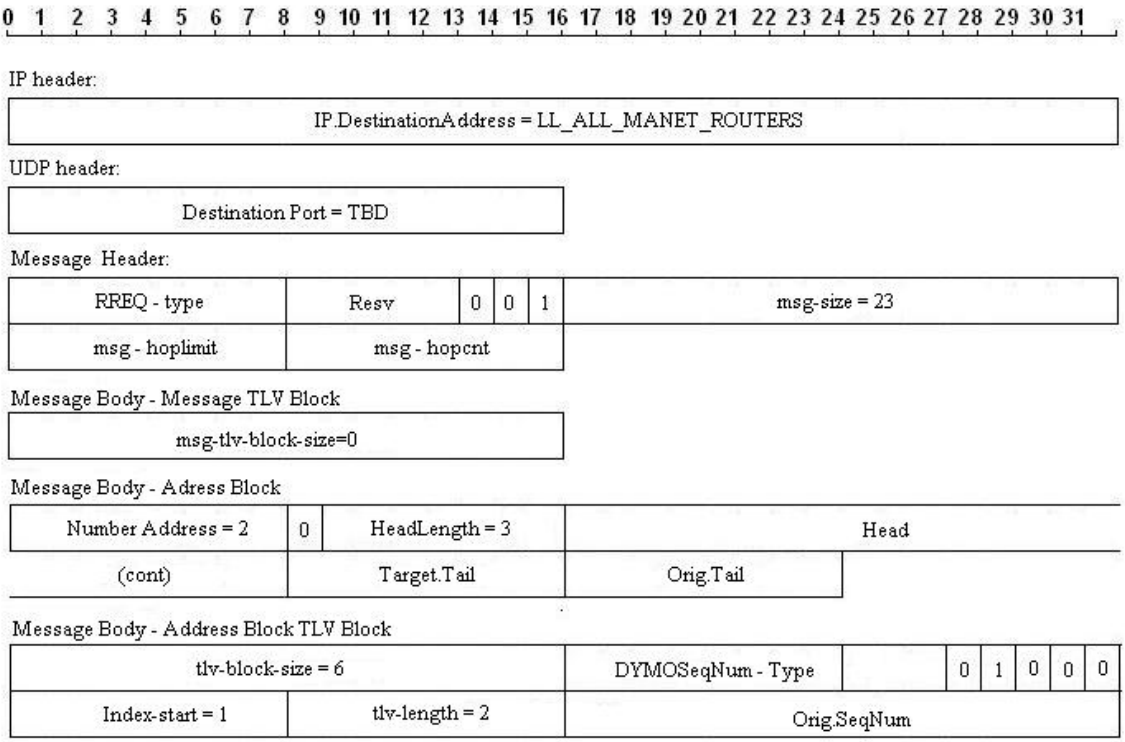
Un Routing message contiene la siguiente información:

- IP.DestinationAddress
- UDP.DestinationPort
- MsgHdr.HopLimit
- AddBlk.OrigNode.Address
- AddTLV.OrigNode.SeqNum

Un Routing Message puede incluir opcionalmente los siguientes campos:

- AddTLV.TargetNode.SeqNum
- AddTLV.TargetNode.HopCnt
- AddBlk.AdditionalNode.Address
- AddTLV.AdditionalNode.SeqNum
- AddTLV.Node.HopCnt
- AddTLV.Node.Prefix

Ejemplo de IPv4 RREQ:



3.3.2.4.1.2.3. Route Error (RERR)

Un mensaje RERR se usa para propagar la información de que una ruta no está disponible para una o más direcciones IP.

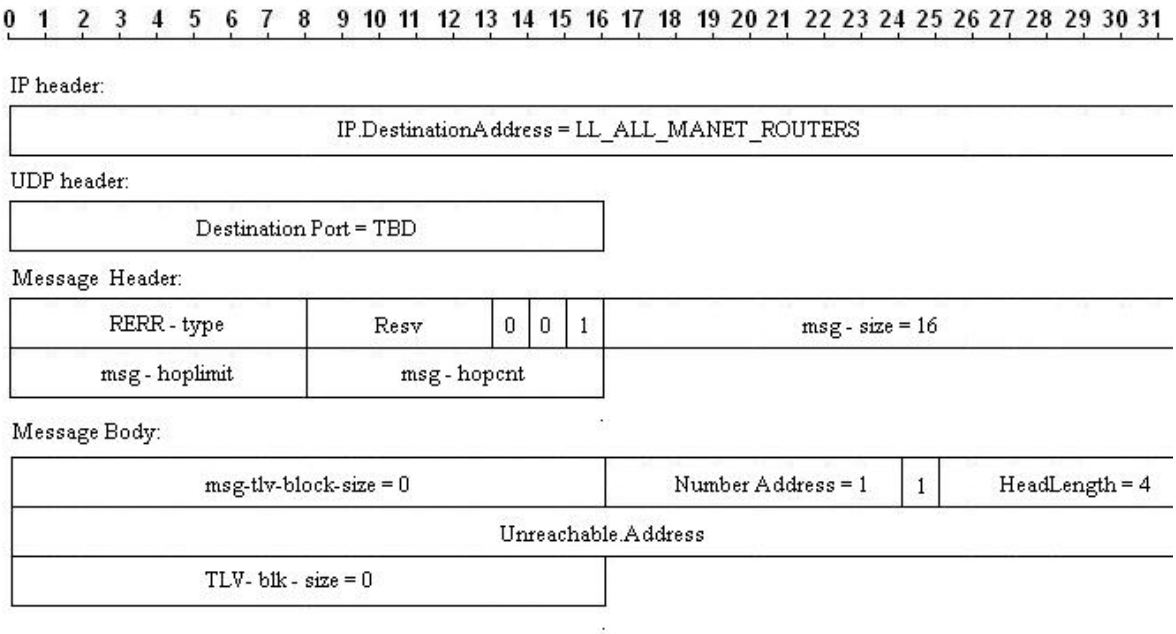
Un RERR requiere la siguiente información:

- IP.DestinationAddress
- UDP.DestinationPort
- MsgHdr.HopLimit
- AddBlk.UnreachableNode.Address

Campos opcionales:

- AddTLV.UnreachableNode.SeqNum

Ejemplo IPv4 RERR:



3.3.2.4.1.3. Números de secuencia

El protocolo DYMO requiere que cada nodo de la red preserve su número de secuencia (*OwnSeqNum*) para asegurarse un mantenimiento de la red libre de bucles. Estos números de secuencia permiten que los nodos determinen el orden de descubrimientos de rutas, con lo que no permiten el uso de información caducada. Hay diferentes números de secuencia:

- El número de secuencia de la dirección del nodo destino de la tabla de encaminamiento de un nodo (*Route.SeqNum*).
- El número de secuencia del nodo destino cuando se envía un RE (*TargetSeqNum*). Si no tiene ningún número en su tabla, el valor que debe llevar es el de cero.
- El número de secuencia del nodo del bloque que transporta el RE (*RBNodeSeqNum*).
- El número de secuencia del nodo inalcanzable debido a que la ruta se ha roto (*UNodeSeqNum*), es transportado por el paquete RERR, en el caso que lo desconozca este será cero.

Para incrementar el OwnSeqNum se tiene que crear un RREQ o un RREP, y cumplir una de las dos condiciones siguientes:

- Que el TargetNumSeq sea superior al número de secuencia del nodo (*OwnSeqNum*).

- Que el TargetNumSeq sea igual al OwnSeqNum y el número de nodos por los que ha pasado un RE es menor al número de nodos intermedios por los cuales ha pasado un bloque hasta llegar al nodo.

3.3.2.4.1.4. Entradas en la tabla de rutas

La tabla de rutas es actualizada cuando se reciben paquetes de control del protocolo DYMO. Estos paquetes pueden contener información de una nueva ruta, actualización sobre un enlace o pueden indicar la ruptura de un enlace. Cuando se recibe un RE y no se tiene constancia de la ruta especificada en el paquete, el nodo crea una nueva ruta. El nodo introduce en la tabla de enrutamiento los siguientes datos:

- La dirección IP del nodo destino.
- El número de saltos que hay entre el emisor y dicho nodo.
- El tiempo de vida de la ruta (*ROUTE_TIMEOUT*).
- La dirección del siguiente nodo hacia el destino.
- La interfaz por la cual reenvía los datos.
- El tamaño de la subred.
- El número de secuencia del nodo destino.
- Un indicador para saber si el nodo actúa como gateway (Es un nodo en una red informática que sirve de punto de acceso a otra red).

En el caso de que una ruta exista en la tabla de encaminamiento se tiene que actualizar cuando:

- El temporizador no haya expirado y el número de saltos del paquete de control sea superior o igual al número que el nodo tiene en su tabla de encaminamiento.
- El temporizador ha expirado y el número del paquete es uno más que el que se tiene en la tabla.
- El número de secuencia del RE es superior que el que tiene el nodo.
- El número de secuencia del nodo destino en la tabla de enrutamiento no es conocido y se recibe un RE con un número de secuencia para ese destino.

Un nodo deja inactiva una ruta después de que pase un tiempo ROUTE_TIMEOUT, este temporizador se va reiniciando cada vez que el nodo reciba un paquete de datos y contenga esa ruta, ya que indica que está utilizando el enlace. Una vez que llegue a cero el temporizador ROUTE_TIMEOUT, la ruta pasa a estado inactivo durante un ROUTE_DELETE_PERIOD. Entonces, ésta no se puede utilizar y es borrada por el nodo cuando expira el temporizador.

Cuando un nodo indica la ruptura de un enlace o le llega un paquete RERR indicando la ruptura de un enlace que utilizaba, el nodo tiene que poner el temporizador ROUTE_TIMEOUT de la tabla de encaminamiento de dicho nodo al final, así se obliga a que la ruta pase a estado inactivo.

3.3.2.4.2. Funcionamiento

El funcionamiento del protocolo DYMO se divide en dos mecanismos básicos; descubrimiento de ruta y mantenimiento de ruta. En el primero se explica cómo se puede descubrir una ruta en la red, y en el segundo se expone la forma que tiene el protocolo para detectar rupturas en las rutas. De esta manera los nodos descubren otra forma de llegar al destino.

3.3.2.4.2.1. Descubrimiento de ruta

Siempre que un nodo intenta enviar un paquete a un destino, el emisor comprueba que el destino esté en la tabla de encaminamiento. En el caso que ya exista esta ruta, el paquete envía la información al siguiente nodo basándose en la tabla. En el caso de que no esté, se realiza el mecanismo de descubrimiento de ruta.

Este mecanismo envía en modo broadcast el paquete RE indicando en un flag, que se trata de un mensaje de control RREQ. Cuando se crea este mensaje el OwnSeqNum se tiene que incrementar en una unidad. Los principales datos de este mensaje de control son:

- El TTL indica el número de saltos que le faltan para desechar el paquete, cuando es creado el valor es NET_DIAMETER.
- La dirección del nodo a la que se quiere enviar datos.
- El número de secuencia del nodo destino, en el caso que no se conozca este valor está a cero.
- Número de saltos por el que ha pasado el paquete (*THopCnt*).
- Estructuras de datos que informan del encaminamiento de una dirección (RBlock).

Cuando se crea un RREQ se tiene que crear el primer RBlock, el cual contiene:

- La dirección del nodo del RBlock al que pertenece.
- El número de secuencia del nodo que tiene la dirección en el bloque.
- El número de saltos que ha pasado este bloque.
- Un bit indicando si actúa como gateway.

El nodo que crea el RREQ se tiene que esperar RREQ_WAIT_TIME para poder enviar otro mensaje de RREQ. Para reducir una posible congestión en la red, éste tiene que seguir un tiempo de backoff binario exponencial, es decir, el primer intento de RREQ tiene que esperar el tiempo RREQ_WAIT_TIME por dos, el segundo por cuatro y así sucesivamente. Se pueden hacer hasta

RREQ_TRIES intentos antes de notificar que el nodo no es accesible. En el descubrimiento de la ruta el nodo emisor guarda los paquetes en un buffer, del que se borra la información si se agotan los intentos. En el caso que le llegue un RREQ a un nodo intermedio y éste no disponga de la dirección del nodo destino, o el número de secuencia del RREQ (RBNodeSeqNum) sea superior al de su tabla de encaminamiento (Route.SeqNum), este paquete deberá actualizar la tabla de encaminamiento del nodo para realizar la ruta inversa. El nodo tiene que actualizar su OwnSeqNum sumándole uno, introducir un nuevo RBlock con sus propios valores, disminuir el valor del TTL y sumarle un salto al campo THopCnt. Una vez introducidos los cambios en el paquete se reenvía en modo broadcast. Si un nodo recibe un paquete RREQ que contenga la dirección destino en su tabla de encaminamiento, entonces compara si el Route.SeqNum es superior al TargetSeqNum del paquete. En caso que no lo sea, se reenvía, ya que el nodo tiene caduca su tabla y actualiza los campos. Cuando es superior el nodo tiene que actualizar su OwnSeqNum sumándose uno y crear un paquete RE indicando que es RREP de respuesta al descubrimiento. Se introducen los saltos que le falta para llegar a dicho nodo, el número de secuencia del nodo destino, la dirección del siguiente salto y el RBlock del nodo que lo emite. Este nodo no debe enviar ningún paquete de RREQ al destino, por lo que el nodo destino tiene que hacer también un descubrimiento de ruta si desea realizar un enlace bidireccional. Por último tenemos el caso que el nodo destino sea el que procesa el RREQ, en este caso el nodo compara si el OwnNumSeq es superior al TargetSeqNum, en el caso que no sea así se descarta el paquete. Contrariamente, si es superior, el nodo guarda los datos del RREQ en la tabla de encaminamiento para un posible enlace bidireccional. El nodo debe crear un paquete RE indicando que es una repuesta a un descubrimiento (*RREP*). Aumenta en uno el OwnSeqNum e introduce los datos en el paquete, este paquete se transmite al último nodo que ha transmitido el RREQ.

Una vez se envía el mensaje RREP por la ruta inversa, los nodos tienen que transmitirlo de forma unicast (Es la comunicación establecida entre un solo emisor y un solo receptor en una red) por la ruta inversa que le ha llegado el RREQ, estos nodos tienen que sumarle uno al OwnSeqNum y actualizar su tabla de enrutamiento. Cuando es recibido por el creador del mensaje RREQ, éste está preparado para transmitir los datos, estos son enviados vía el buffer.

3.3.2.4.2.2. Mantenimiento de las rutas

Este apartado es similar al descrito en el protocolo DSR. Cada nodo es el encargado de mantener el enlace del siguiente nodo. Cuando un nodo detecta la pérdida de un enlace, éste crea un RERR y lo transmite a los nodos anteriores de la ruta. Para el descubrimiento de una ruptura en una ruta, el draft propone cuatro posibles alternativas:

- Reconocimientos en la capa de enlace (*similar al usado en DSR*).
- Mensajes de Hellos.
- Descubrimientos de vecinos.

- Timeout de ruta: Este mecanismo no transmite el RERR. Una vez pasado un tiempo ROUTE_TIMEOUT la ruta queda inhabilitada, este tiempo va actualizándose cada vez que pasa un paquete por el nodo que utiliza dicha ruta.

Cuando un nodo descubre una ruptura, éste tiene que crear un paquete RERR, el cual tiene que introducir en el campo UNodeAddress la dirección del nodo inalcanzable, en el caso que se sepa el número de secuencia del nodo se introduce en el campo UNodeSeqNum. En caso que no se conozca, este campo toma valor cero. En el campo TTL se introduce el NET_DIAMETER y se envía en modo broadcast, si hay otra ruta que es perjudicada por la ruptura del enlace, se introduce también en el paquete RERR, en el cual se añaden nuevos campos de dirección y de números de secuencia.

Cuando un nodo recibe un RERR tiene que invalidar la ruta si:

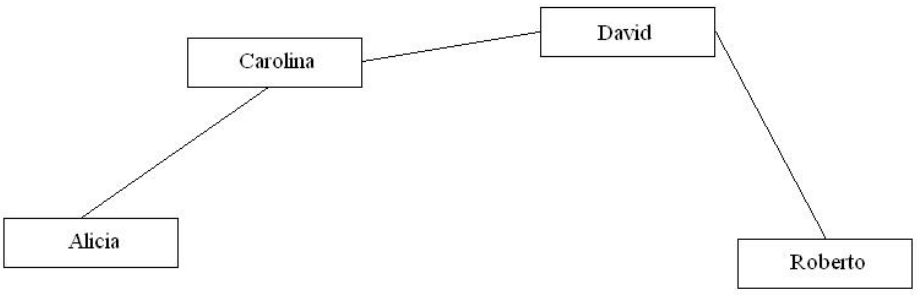
- La ruta que invalida tiene como siguiente salto la misma dirección IP del paquete que ha transmitido el RERR.

- La ruta que invalida tiene como siguiente salto la interfaz del paquete que ha transmitido el RERR.

- El número de secuencia del nodo que nos ha llegado es cero o el resultado de restar el Route.SeqNum del nodo destino y el número de secuencia de la ruta inalcanzable del paquete es menor o igual a cero.

Si no pasa por algún filtro de éstos, la ruta inválida se retira del paquete y si el RERR contiene alguna ruta más se reenvía en modo broadcast. En caso que el paquete ya no tenga ninguna ruta inválida, éste ya no se transmitirá más. Si el paquete pasa todos los filtros, el nodo tiene que reenviarlo en modo broadcast bajando el campo TTL.

Ejemplo



1. Paso 1

- **Alicia** quiere intercambiar datos con **Roberto**
- **Alicia** no conoce un camino a **Roberto** todavía, así que envía un nuevo RREQ de una ruta a Roberto.

2. Paso 2
   - Carolina recibe el RREQ de Alicia, refresca la información almacenada de como llegar directamente a **Alicia**, después añade información sobre si mismo al paquete y lo reenvía.
3. Paso 3
   - David recibe el RREQ de Carolina, refresca la información almacenada de como llegar a Carolina directamente y a **Alicia** (vía Carolina), después añade información sobre si mismo al paquete y lo reenvía.
   - Al mismo tiempo, **Alicia** recibe el RREQ de Carolina. Tras examinar el contenido del paquete, llega a la conclusion de que no es útil connocer la información de que como contactar consigo misma, así que descarta el paquete.
4. Paso 4
   - Roberto recibe el RREQ de David y refresca a información de cómo contactar con David directamente, con Carolina (vía David) y con **Alicia** (vía David). Detecta que é les el destinatario del RREQ, así que crea un RREP con Alicia como destinataria., y sabiendo que puede contactar con ella vía David, envía el mensaje a éste último.
   - Al mismo tiempo, Carolina recibe el RREQ de David, pero lo ignora.
5. Paso 5
   - David recibe el RREP a **Alicia** enviado por **Roberto**, refresca la información de como llegar directamente a **Roberto**, añade sus datos al mensaje, y sabiendo que puede llegar a **Alicia** vía Carolina, lo manda a Carolina.
6. Paso 6
   - Carolina recibe el RREP a **Alicia** enviado por David, refresca la información de como llegar directamente a David y a **Roberto** (via David), añade sus datos al mensaje, y sabiendo que puede llegar a **Alicia** directamente, lo envía a **Alicia**.
7. Paso 7
   - **Alicia** recibe el RREP que le ha enviado Carolina y refresca la información de cómo llegar a ella directamente, a David (vía Carolina) y a **Roberto** (vía Carolina). Ahora que sabe como contactar con **Roberto** ella puede enviar los paquetes para él a Carolina.
8. Paso 8
   - Carolina recibe el paquete de datos para **Roberto** de **Alicia**. Como sabe que David puede llegar a **Roberto** le manda el mensaje.
9. Paso 9
   - David recibe el paquete de datos para **Roberto**. Como sabe que puede llegar directamente a **Roberto**, le manda el paquete.
10. Paso 10
    - **Roberto** recibe el paquete de datos. Como todavía continua sabiendo como llegar a **Alicia**, el podría responder con otro mensaje, pero decide no hacerlo.

3.3.2.4.3. Parámetros y valores por defecto para DYMO

Los valores representados en la tabla siguiente pueden ser configurados dependiendo de la amplitud de la red o el dinamismo de topología de una red. Por ejemplo, en una red donde los nodos no sufran cambios en su ubicación y sea estable, el ROUTE_TIMEOUT tiene que ser mayor que el descrito en el draft. Todos los nodos deben llevar los mismos ajustes de parámetros, ya que un desajuste en parámetros en el ROUTE_TIMEOUT o en el ROUTE_DELETE_TIMEOUT pueden producir bucles en la red o roturas de un enlace, debido a retrasos arbitrarios de paquetes. Los principales valores que se tienen que seguir para realizar una correcta implementación del protocolo DYMO son los descritos en la siguiente tabla.

Valores por defecto del protocolo DYMO

Nombre de la variable	Cantidad	Unidades
NET_DIAMETER	10	Saltos
ROUTE_TIMEOUT	10	Mbps
RATE_LIMIT	3000	Milisegundos
ROUTE_DELETE_TIMEOUT	5*ROUTE_TIMEOUT	Milisegundos
RREQ_WAIT_TIME	1000	Milisegundos
RREQ_TRIES	3	Intentos
ROUTE_DELETE_PERIOD	-	-
UNICAST_MESSAGE_SENT_TIMEOUT	-	-
TTL	NET_DIAMETER	Saltos

3.3.2.5. DYMOLoW, una adaptación a redes LoWPAN.

3.3.2.5.1. Red LoWPAN

Una LoWPAN es una red de comunicaciones de bajo coste que permite conectividad sin hilos a elementos de funcionalidades reducidas en aplicaciones sin grandes requerimientos de velocidad. Estas redes cumplen el estándar IEEE 802.15.4, del cual se habla en el siguiente apartado. Ofrecen soporte para direcciones de 16 o 64 bits, su ancho de banda es reducido (250 kbps, 40 kbps y 20 kbps para las capas físicas definidas de 2.4 GHz, 915 MHz y 868 MHz respectivamente) y permite topologías mesh o en estrella. Están orientadas a dispositivos como los sensores: de bajo coste, con capacidades de procesado, memoria de almacenamiento y batería limitados. En el ámbito de las LoWPANs existen varios problemas y requerimientos:

- *Tamaño limitado de paquete*: el máximo tamaño de paquete en capa física es de 127 bytes, y en capa MAC de 102 bytes.

- *Conectividad IP*: el gran número de dispositivos plantea la necesidad de un espacio direccionamiento grande. Además sería aconsejable una autoconfiguración sin estado y solucionar la interconexión con otras

redes IP, incluyendo Internet. IPv6 propone soluciones a estos problemas, pero hay que tener en cuenta el limitado tamaño de paquete.

- *Topologías*: dado que las topologías mesh implican encaminamiento multisalto, el protocolo de encaminamiento debe minimizar el overhead así como los requerimientos de memoria y procesado.

- *Gestión y configuración limitada:* puede darse el caso en el que dada la localización de algunos dispositivos éstos sean difíciles de acceder. Por este motivo, la configuración de los protocolos debería ser mínima.

- *Descubrimiento de servicios*: las LoWPANs requieren protocolos simples de descubrimiento para descubrir, controlar y mantener los servicios que proporcionan los dispositivos.

- *Seguridad*: debe considerarse dependiendo de las necesidades de cada aplicación. IEEE 802.15.4 proporciona seguridad a nivel de enlace mediante AES (Advanced Encryption Standard). Debido a la naturaleza de los dispositivos, deberían descartarse soluciones que impliquen un excesivo ancho de banda o nivel de cálculo.

Dadas las características y limitaciones de las redes LoWPAN en cuanto a duración de baterías, memoria y capacidad de procesado de los dispositivos, así como los frecuentes cambios en la topología, el encaminamiento en este tipo de redes es un punto crítico. Los protocolos deberán adaptarse a estas características ya que de otro modo el encaminamiento no sería eficiente ni adecuado.

3.3.2.5.2. Protocolos actuales.

Uno de los objetivos retos actuales es adaptar los protocolos de MANET a las redes LoWPAN. Actualmente existen dos especificaciones en formato draft en vía de estandarización en este ámbito: el DYMO-Low , adaptación del protocolo DYMO, y el LOAD, adaptación de AODV. DYMO-Low especifica cómo usar el protocolo DYMO de MANET en redes 802.15.4 para proporcionar un encaminamiento mesh. Crea una capa y también la topología de la red mesh por debajo de IP, de manera que IP ve la PAN como un solo enlace. En DYMO-Low los mensajes RREQ son broadcast y sólo son pueden ser respondidos por el destino. En el caso más simple para seleccionar la mejor ruta se utiliza el coste acumulativo de saltos, pero se sugiere emplear otro criterio que tenga en cuenta la calidad del enlace por medio del indicador LQI. Para determinar si un vecino es accesible no se hace uso de los mensajes de *hello* sino de un mecanismo de reconocimiento. Se utilizan números de secuencia (el campo se ha reducido de 32 a 8 bits) para evitar bucles y se permite el envío de múltiples elementos de encaminamiento en un mismo paquete de control para ahorrar energía.

Por su parte LOAD especifica cómo usar el protocolo AODV de MANET en redes 802.15.4. Está definido para operar encima de la capa de adaptación en lugar de la capa de transporte, creando una red mesh por debajo de la capa

IPv6. En LOAD sólo el destinatario de una ruta puede generar la respuesta RREP en el descubrimiento, y no se hace uso de números de secuencia. La reparación local de rutas es opcional, y la métrica se basa en el uso del LQI.

3.3.2.6. Implementaciones del DYMO

Son muy pocas las implementaciones disponibles para este protocolo ya que es bastante reciente.

- NIST-DYMO (*S.O:Linux*)
  - Creado por el National Institute of Standards and Technology, funciona sobre Linux con kernels 2.4 y 2.6.
  - Es necesario el modulo NETFILTER para el correcto funcionamiento de la implementación.
  - Da soporte para múltiples interfaces, puede hacer funciones de Gateway y puede hacer el enrutado de subredes locales.
  - El descubrimiento para que sea bidireccional se tiene que realizar en los dos extremos.
  - Homepage:http://wwwx.antd.nist.gov/twiki/bin/view/ANTDProjects/NistDymo
  - Parámetros del NIST-DYMO:

Nombre de la variable	Valor	Unidades
NET_DIAMETER	10	Nodos
HELLO_LOSS	4	Paquetes
HELLO_GAIN	7	Paquetes
HELLO_WINDOW	10	Paquetes
HELLO_INTERVAL	1	Segundo
RREQ_WAIT_TIME	1	Segundo
ROUTE_TIMEOUT	3	Segundos
ROUTE_DELETE_TIMEOUT	5 * ROUTE_TIMEOUT	Segundos

- DYMO-UM (S.O:Linux)
  - Creado por Pedro M. Ruiz y Francisco Ros, de la Universidad de Murcia.
  - Funciona en plataformas linux con kernels 2.4, 2.6 y sobre el simulador de redes NS-2

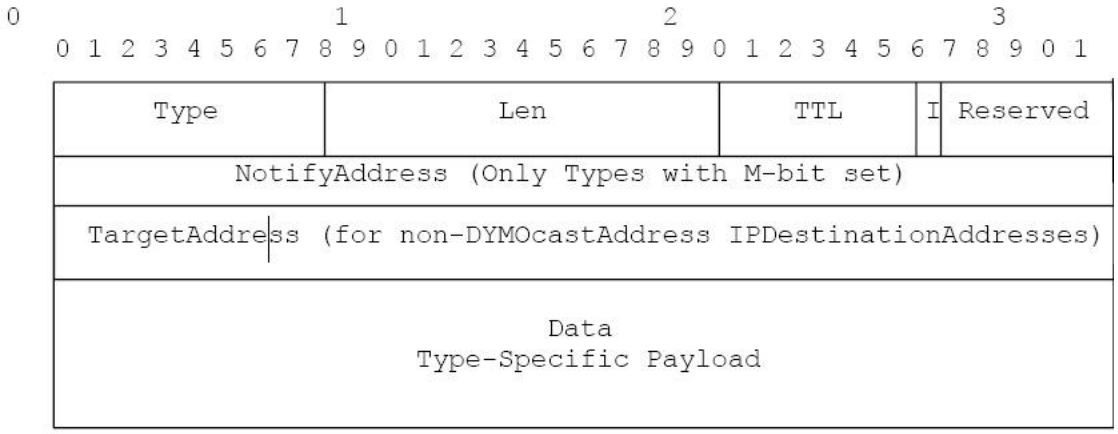
- Escrito en lenguaje de programación C y C++.
- Es necesario el modulo integrado en Linux NETFILTER.
- La versión 6 del protocolo IP no está disponible.
- Homepage:
  http://masimum.dif.um.es/?Software:DYMOUM:Introduction
- Parámetros de DYMOUM:

Nombre de la variable	Valor	Unidades
Dymo_RateLimit	10	Mbps
Net_Diameter	10	Nodos
Route_Timeout	3	Segundos
Route_Delete_Timeout	5 * Route_Timeout	Segundos
Dymo_Max_NR_Interfaces	10	Interfaces

Tal y como se puede observar en las implementaciones, el protocolo Dymo es un protocolo reciente que se está en continuo desarrollo para que en un futuro cercano se realice el estándar y pueda implantarse en las redes Ad-Hoc del futuro.

3.3.2.7. Formato de los paquetes del protocolo DYMO

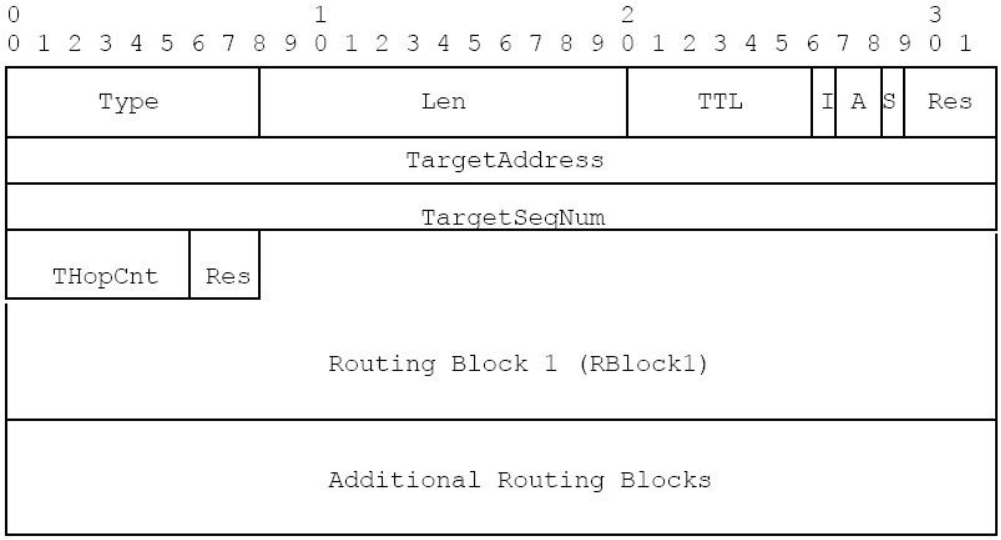
Generic Dymo Element Structure



- Type: El campo de Tipo identifica el elemento.
- Len: el campo que indica el tamaño del elemento en bytes, incluyendo la parte fija.
- TTL: el campo que identifica el número máximo de veces el elemento debe ser transmitido de nuevo.
- I bit: Si esta activo algún nodo no ha hecho caso a este paquete.

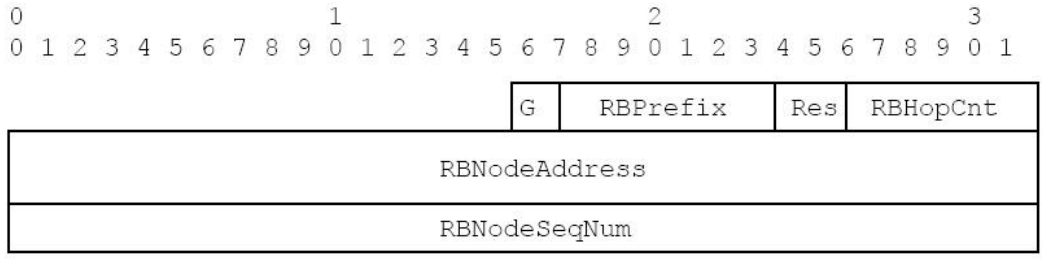
- Reserver: Estos bits están reservados, en caso normal irán a 0.
- NotifyAddress: Se introduce el nodo para enviar un UERR si el tipo de elemento no es soportado.
- TargetAddres: la dirección de nodo que esta destinada el elemento.
- Data:datos.

Routing Element



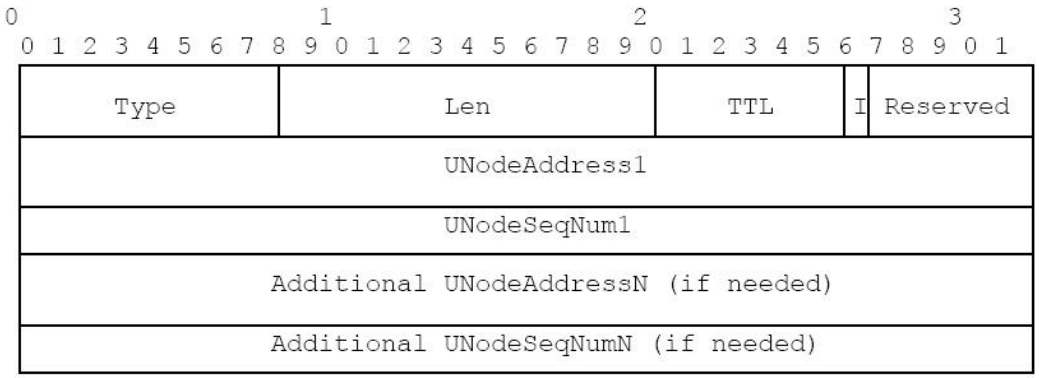
- A bit: El selector que indica si este RE requiere un RREP por el TargetAddress.
- S bit: El selector que indica si este RE requiere enviar un mensaje unicast a la dirección de salto anterior.
- Target Address: El nodo que es la última destinación del RE.
- TargetSeqNum: El número de secuencia de la última destinación de este RE. Si el Número de Secuencia es desconocido para esta Ruta entonces se pone a cero.
- THopCnt: el campo que identifica el número de nodos intermedios por el que un paquete es atravesado a lo largo de una ruta.
- RBlock: La estructura de datos que describe la información de encaminamiento relacionada con una dirección IP particular, RBNodeAddress.

Routing Block



- G bit: el selector para indicar si el RBNODEADDRESS es un gateway. Si G=1 RBNodeAddress es un gateway.
- RBPrfix: El campo que especifica el tamaño de la subred accesible por el nodo asociado.
- RBHopCnt: El campo que identifica el número de nodos intermedios por el que ha pasado el RBLOCK asociado.
- RBNodeAddress: La dirección IP asociada a este bloque.
- RBNodeSeqNum: El número de secuencia del nodo asociado a este RBLOCK.

Route Error



- UNodeAddress1: La dirección de IP del nodo inalcanzable.
- UNodeSeqNum1: El número de secuencia del nodo inalcanzable, si no se sabe se pone a cero.

3.3.3. AODV (Ad Hoc On-Demand Distance Vector)

3.3.3.1. Visión general

El protocolo de encaminamiento Ad Hoc On-Demand Distance Vector (AODV) es un protocolo de encaminamiento reactivo para redes ad hoc que mantiene las rutas solamente entre los nodos que necesitan comunicarse. Los mensajes de encaminamiento no contienen información sobre la trayectoria completa de la ruta, sino solamente sobre la fuente y el destino. Por lo tanto,

los mensajes de encaminamiento tienen un tamaño fijo. Utiliza números de secuencia de destino para especificar cuán fresca o actual es una ruta (en relación a otra), el cual también se utiliza para evitar bucles.

3.3.3.2. Funcionamiento

Cuando un nodo fuente desea enviar un mensaje a un nodo destino para el cual no tiene una ruta válida, es decir, una entrada de ruta para el destino la cual tenga asociado un número de secuencia mayor o igual que el contenido en cualquier RREQ que el nodo ha recibido para ese destino, inicia un proceso de descubrimiento de ruta. El nodo fuente difunde un mensaje RREQ a sus vecinos, los cuales entonces reenvían la solicitud a sus vecinos y así sucesivamente, hasta que es alcanzado el destino o un nodo intermedio con una ruta al destino en su tabla de encaminamiento (figura 1). Durante el proceso de reenvío de RREQ, un nodo intermedio registra en su tabla de encaminamiento (por ejemplo, una lista de precursores) la dirección del vecino del cual recibió la primera copia del mensaje de broadcast, de ese modo establece una ruta hacia atrás. Las copias adicionales del mismo RREQ recibidas luego son descartadas. Una vez que el RREQ llegó al destino o nodo intermedio con una ruta, el respectivo nodo responde haciendo un unicast del mensaje Route REPly (RREP) hacia el vecino del cual recibió el primer RREQ, el cual transmite el RREP hacia atrás a través de los nodos precursores hacia el nodo fuente (figura 2). Es importante observar que la única información mutable en un RREQ y en un RREP es el contador de salto (que está siendo incrementado en cada salto). En el caso que el RREQ es respondido por un nodo intermedio (y si el RREQ tenía establecida esta opción), el nodo intermedio también envía un RREP al destino. De esta manera, se permite establecer la trayectoria de ruta bidireccionalmente. En el caso que un nodo recibe una nueva ruta (mediante un RREQ o mediante un RREP) y el nodo tiene ya una ruta 'tan fresca' como la recibida, la más corta será actualizada.

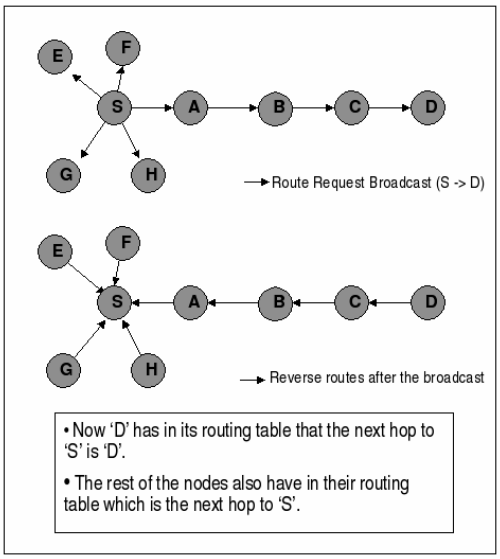


Figura 1. Route Request

Si existe una subred (una colección de nodos identificados por prefijo de red común) que no utiliza AODV como su protocolo de encaminamiento y desea poder intercambiar información con una red AODV, uno de los nodos de la subred puede ser seleccionado como su "líder de red". El líder de red es el único nodo de la subred que envía, reenvía y procesa mensajes de encaminamiento AODV. En cada RREP que el líder emita, establece el tamaño del prefijo de subred.

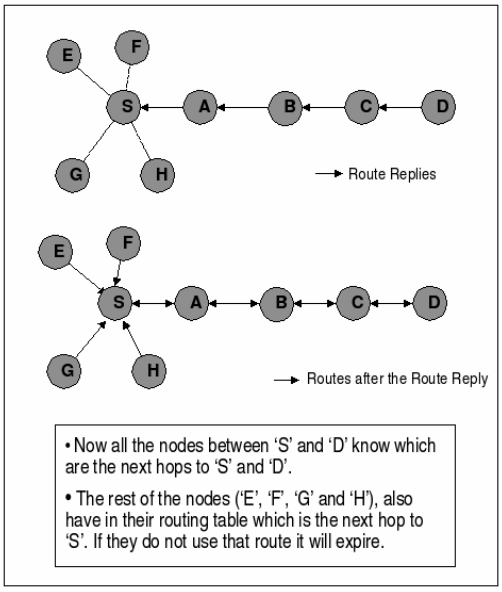


Figura 2. Route Reply

Opcionalmente, un mensaje del Route REPly ACKnowledgment (RREP-ACK) puede ser enviado por el creador del RREQ para reconocer la recepción del RREP. El mensaje de RREP-ACK no tiene ninguna información mutable.

Además de estos mensajes de encaminamiento, el mensaje Route ERRor (RERR) se utiliza para notificar a otros nodos que ciertos nodos ya no están más accesibles debido a un enlace roto (figura 3). Cuando un nodo vuelve a difundir (broadcast) un RERR, agrega solamente los destinos inalcanzables a las cuales el nodo podía reenviar mensajes. Por lo tanto, la información mutable en un RERR es la lista de destinos inaccesibles y el contador de los destinos inalcanzables incluidos en el mensaje. De todas formas, es previsible que, en cada salto, la lista de destinos inalcanzables pueda no cambiar o convertirse en un subconjunto de la original.

Comentario [NBT1]: Se descarta el RERR recibido y se crea otro nuevo RERR que se reenvía o se modifica y se reenvía? SEGÚN RFC 3561 PAG 25, tesis pag 27 . .

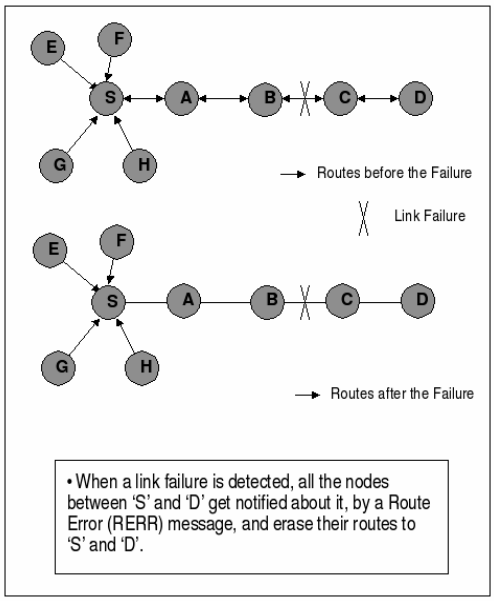


Figura 3. Route Error

Las rutas se mantienen de la siguiente forma: mensajes HELLO son enviados periódicamente mediante un broadcast hacia los nodos vecinos. Cuando un nodo fuente se mueve, tiene que reiniciar el proceso de descubrimiento de ruta para hallar una nueva ruta hacia el destino. Por otro lado, cuando un nodo intermedio a través de la ruta se mueve, sus vecinos en sentido ascendentes(a partir de ellos hacia el origen) informarán un enlace roto debido al movimiento y propagarán un mensaje RERR a cada uno de sus vecinos activos en sentido ascendente (a partir de ellos hacia la fuente). Estos nodos a su vez propagan el paquete RERR a sus vecinos ascendentes, y así hasta llegar al nodo fuente. El nodo fuente puede decidir si reiniciará el descubrimiento de ruta para ese destino si aún desea una ruta hacia él.

Es necesario aclarar que el único campo mutable en los paquetes de encaminamiento es el Contador de Saltos. Las figuras del Apéndice A muestran la estructura de los mensajes AODV e indican cuáles son los campos mutables.

3.3.3.3. Vulnerabilidades

Dado que AODV no tiene ningún mecanismo de seguridad, los nodos maliciosos pueden realizar muchos ataques solamente con no cumplir las reglas de AODV. Un nodo malicioso *M* puede realizar los siguientes tipos de ataques (entre muchos otros) contra AODV:

1. Hacerse pasar por un nodo *S* mediante la falsificación de un RREQ con su dirección como la dirección del creador.

2. Cuando se reenvía un RREQ generado por *S* para descubrir una ruta a *D*, reducir el campo contador de salto para aumentar las posibilidades de estar en

la trayectoria de la ruta entre *S* y *D* así poder analizar la comunicación entre ellos. Una variante de esto es incrementar el número de secuencia de destino para hacer que los otros nodos creen que esta es una ruta más 'fresca'.

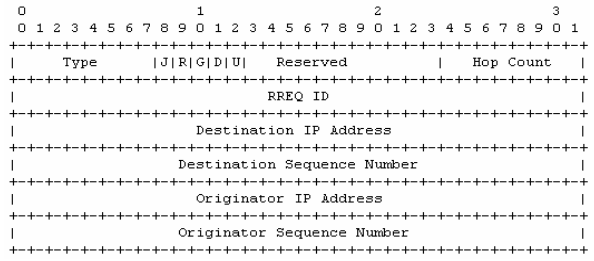
3. Hacerse pasar por un nodo *D* mediante la falsificación de un RREP con su dirección como la dirección del destino.

4. Hacerse pasar por un nodo falsificando un RREP que afirma que él es el nodo destino y, para aumentar el impacto del ataque, afirmar que es un líder de red de una subred *SN* con un número de secuencia grande y enviarlo a sus vecinos. De esta manera se convertirá (por lo menos localmente) un agujero negro (black hole) para la subred *SN* entera.
5. Selectivamente, no reenviar ciertos RREQs y RREPs , no responder a ciertos RREPs y no reenviar ciertos mensajes de datos. Esta clase de ataque es especialmente difícil de detectar incluso porque los errores de transmisión tienen el mismo efecto.

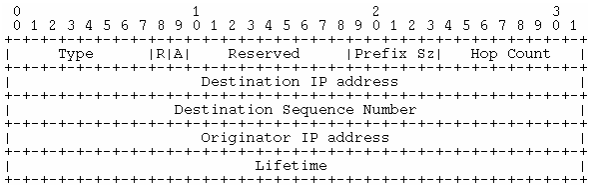
6. Falsificar un mensaje RERR pretendiendo ser el nodo *S* y enviarlo a su vecino *D*. El mensaje RERR tiene un número de secuencia de destino muy alto *dsn* para una de los destinos inalcanzables (*U*). Esto podría causar que *D* actualice el número de secuencia destino que correspondía a *U* con el valor *dsn* y, por lo tanto, los descubrimientos de ruta futuros realizados por *D* para obtener una ruta a *U* fallarán (porque el número de secuencia destino de *U* será mucho más pequeño que el que está almacenado en la tabla de encaminamiento de *D*).

7. Según el draft actual de AODV, el creador de un RREQ puede poner un número secuencia destino mucho más grande que el verdadero. Además, los números de secuencia wraparound cuando alcanzan el valor máximo permitido por el tamaño de campo. Esto permite un ataque muy fácil en donde un atacante es capaz de fijar el número de secuencia de un nodo a cualquier valor deseado apenas enviando dos mensajes de RREQ al nodo.

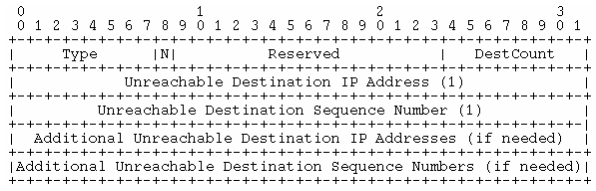
3.3.3.4. Formato de mensajes AODV



Formato de Mensaje Route Request (RREQ)
Campo mutable: Hop Count

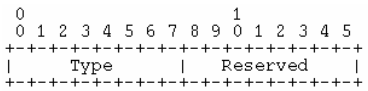


Formato de Mensaje Route Reply (RREP)
Campo mutable: Hop Count



Formato de Mensaje Route Error (RERR)
Campo mutable: Ninguno

Comentario [NBT2]: un RERR no tiene campos mutables que deban ser protegidos, como los tienen el RREQ o el RREP



Formato de Mensaje Route Reply Acknowledgment (RREP-ACK)
Campo mutable: Ninguno

3.3.4. SAODV

3.3.4.1. Visión General

El protocolo Secure Ad Hoc On-Demand Distance Vector (SAODV) es una extensión del protocolo de encaminamiento AODV. SAODV es usado para proteger el mecanismo de descubrimiento de ruta proporcionando características de seguridad tales como integridad, autenticación y no repudio. Utiliza como sistema de administración de claves el sistema de Administración de Claves Ad Hoc Simple (SAKM – **ver sección 3.3.4.5**). Fue desarrollado por Manel Guerrero Zapata de la Universidad Politécnica de Catalunya, actualmente es un draft que se encuentra en la versión 6 de Septiembre de 2006.

La idea general de SAODV consiste en que cada nodo tiene los certificados de clave pública de todos los otros nodos de la red, puesto que los nodos intermedios validan todos los paquetes de encaminamiento que se encuentran en tránsito. La operación básica es que el creador del mensaje de control agregue una firma digital, por ejemplo RSA, más el último elemento de una cadena hash, por ejemplo el resultado de aplicar n veces consecutivamente una función hash a un valor aleatorio. Como este mensaje atraviesa la red, los nodos intermedios validan criptográficamente la firma y el valor hash, luego generan el k-ésimo elemento de la cadena hash, con k siendo el número de saltos atravesados, y lo colocan en el paquete. Las respuestas de ruta que son provistas tanto por el destino final como por los

nodos que tienen una ruta activa hacia el destino buscado también son firmados digitalmente.

3.3.4.2. Requerimientos de seguridad que satisface

En la mayoría de los ámbitos el principal servicio de seguridad es la **autorización**. El encaminamiento no es la excepción. Típicamente, un "router" necesita tomar dos tipos de decisiones de autorización. Primero, cuando recibe una actualización de encaminamiento desde el exterior, el "router" necesita decidir si, en consecuencia, modifica su base de información de encaminamiento actual. Esto puede ser definido como una operación de *importar autorización*. En segundo lugar, un "router" puede *exportar autorización* cuando recibe una solicitud de información de encaminamiento. El servicio crítico es el de importar autorización.

La autorización puede requerir otros servicios de seguridad tales como **autenticación** e **integridad**, los cuales suelen utilizar las técnicas de firmas digitales y códigos de autenticación de mensajes (MAC).

Sin embargo, en el contexto del encaminamiento, la **confidencialidad** y el **no repudio** no son necesariamente servicios críticos, aunque el no repudio podría ser de utilidad en aquellas redes ad hoc donde se requiere aislar "routers" que presentan un mal comportamiento, de la siguiente forma: un "router" A el cual recibió un "mensaje erróneo" de otro "router" B puede usar este mensaje para convencer a otros "routers" de que el "router" B no se está comportando correctamente.

El problema de **nodos comprometidos** no es contemplado por este protocolo ya que es solamente crítico en escenarios militares. La **disponibilidad** también está afuera del alcance del protocolo. Aunque sería deseable, parece no ser factible prevenir los ataques por **denegación de servicio** en una red que utiliza tecnología inalámbrica (cuando un atacante puede centrarse en la capa física sin preocuparse por el estudio de los protocolos de encaminamiento).

Por lo tanto SAODV cubre los siguientes requerimientos de seguridad:

- **Importar autorización**: la última autoridad acerca de los mensajes de encaminamiento respecto a cierto nodo destino es ese nodo en sí mismo. Por lo tanto, se autorizará solamente información de ruta en la tabla de encaminamiento si esa información de ruta concierne al nodo que está enviando la información. De esta manera, sin un nodo malicioso miente acerca de él, lo único que causará es que otros no podrán encaminar paquetes al nodo malicioso.

- **Autenticación de fuente**: los nodos requieren ser capaces de verificar que un nodo es quien afirma ser.

- **Integridad**: los nodos requieren ser capaces de verificar que la información de encaminamiento que está siendo enviada ha llegado inalterada.

Los últimos dos servicios de seguridad combinados constituyen la **autenticación de datos**.

3.3.4.3. Funcionamiento

SAODV asume que existe un subsistema de administración de claves que hace posible que cada nodo ad hoc pueda obtener las claves públicas de los otros nodos de la red. Asimismo cada nodo ad hoc es capaz de verificar de manera segura la asociación entre la identidad de un nodo dado y la clave pública de ese nodo. Esto se lleva a cabo a través de un esquema de administración de claves.

Se utilizan dos mecanismos para dar seguridad a los mensajes de encaminamiento de AODV: las **firmas digitales** para autenticar los campos no mutables de los mensajes y las cadenas hash (**hash chains**) para asegurar la información del contador de salto (la única información mutable dentro de los mensajes) tanto de los mensajes RREQ como RREP.

La información relativa a las cadenas hash y a las firmas digitales es transmitida con el mensaje AODV como una extensión de éste denominada *Extensión de Firma*.

Los mensajes RERRs son protegidos de una forma diferente ya que ellos contienen una gran cantidad de información mutable. Además, no interesa cuál nodo desencadena un error de ruta y cuáles nodos son los que precisamente lo reenviarán. La única información relevante es que un nodo vecino esta informando que otro nodo no podrá encaminar mensajes hacia ciertos destinos a partir de ese momento. Por lo tanto, cada nodo (que genera o reenvía un mensaje RERR) usa firmas digitales para firmar el mensaje completo y cualquier vecino que reciba el mensaje verifica la firma.

Para reducir los requerimientos de potencia de procesamiento debido al uso de criptografía asimétrica, SAODV realiza lo que se conoce como *verificación retrasada de firmas*, es decir, los nodos reenvían los mensajes de encaminamiento antes de verificarlos. En el caso del descubrimiento de ruta, el nodo necesitará solamente verificar el mensaje RREQ **después** de recibir y de reenviar el RREP correspondiente. Esto evitará que todos los nodos que no se encuentran en la trayectoria seleccionada tengan que verificar mensajes RREQs (con todo el trabajo de cómputo que esto requiere).

Las modificaciones que se realizan a los originales paquetes RREQ, RREP, RERR y RREP-ACK consisten en agregar campos adicionales para incluir las firmas digitales y las cadenas hash, según el caso y también se agregaron dos tipos más de formatos de paquete, RREQ y RREP con

Extensión de Doble Firma. El formato de las Extensiones de Firma de SAODV se muestra en el punto 3.3.4.9

3.3.4.3.1. Cadenas Hash en SAODV

SAODV utiliza **cadenas hash** para autenticar el contador de salto de los mensajes RREQ y RREP de una manera tal que permita a cada nodo que reciba el mensaje (un nodo intermedio o el destino final) verificar que el contador de salto no ha sido modificado (incrementado/decrementado) por un atacante. Esto previene un **ataque del tipo 2 (ver sección 2.2.4)**. Una cadena hash se forma aplicando una función hash unidireccional repetidamente a un valor inicial aleatorio (seed).

Cada vez que un nodo origina un mensaje RREQ o RREP, realiza las siguientes operaciones:

- Genera un número aleatorio (*seed*).
- Establece el campo Max_Hop_Count al valor TimeToLive (de la cabecera IP)

$$Max\_Hop\_Count = TimeToLive$$

- Establece el campo Hash al valor *seed*.

$$Hash = seed$$

- Establece el campo Hash_Function al identificador de la función hash que se usará. Los posibles valores se muestran en la tabla 1

$$Hash\_Function = h$$

- Calcula Top_Hash aplicando hashing a *seed* Max_Hop_Count veces

$$Top\_Hash = h^{Max\_Hop\_Count}(seed)$$

Donde:

h es la función hash
$h^i(x)$ es el resultado de aplicar la función *h* a *x i* veces

- Asimismo, cada vez que un nodo recibe un mensaje RREQ o RREP, realiza las siguientes operaciones para verificar el contador de salto:
- Aplica la función hash *h* (Max_Hop_Count - Hop_Count) veces al valor del campo *Hash*, y verifica que el valor resultante sea igual al valor contenido en el campo Top_Hash.

$$Top\_Hash == h^{Max\_Hop\_Count - Hop\_Count}(Hash)$$

Donde:

a == *b* se lee: verificar que *a* y *b* sean iguales

- Antes de volver a difundir un RREQ o reenviar un RREP, un nodo aplica una vez la función hash al valor Hash en la Extensión de Firma para representar el nuevo salto.

$$Hash = h(Hash)$$

El campo Hash_Function indica cuál función hash se usará para calcular el hash. Los campos Hash_Function, Max_Hop_Count, Top_Hash y Hash son trasmitidos con el mensaje AODV, en la Extensión de Firma.

Todos ellos, excepto el campo Hash son firmados para proteger la integridad. Dado que el campo Hash_Function está firmado, un nodo que reenvía mensajes sólo podrá usar la misma función que utilizó el creador del mensaje, si el nodo no puede verificar un mensaje de encaminamiento porque no soporta la función hash que ha sido seleccionada, elimina el paquete. Esto se explica a continuación.

Valor	Función Hash
0	Reservado
1	MD5HMAC96[14]
2	SHA1HMAC96[15]
3-127	Reservado
128-255	Dependiente de la implementación

Tabla 1. Posibles valores del campo Hash_Function

3.3.4.3.2. Firmas Digitales en SAODV

Las firmas digitales se utilizan para proteger la integridad de los datos no mutables en los mensajes RREQ y RREP. Eso significa que se firma todo menos el campo Hop_Count del mensaje de AODV y el campo Hash de la extensión SAODV.

El problema principal en la aplicación de firmas digitales es que AODV permite a los nodos intermedios contestar a mensajes RREQs si tienen una ruta lo "suficientemente fresca" al destino. Mientras que esto hace más eficiente al protocolo también hace más complicado darle seguridad. El problema es que un mensaje RREP generado por un nodo intermedio debe poder ser firmado por éste a nombre de destino final. Y, además, es posible que la ruta almacenada en el nodo intermedio hubiera sido creada como ruta reversa después de recibir un mensaje RREQ (lo que significa que no tiene la firma para el RREP).

Para resolver este problema, SAODV ofrece dos alternativas. Primero es que, si un nodo intermedio no puede contestar a un mensaje RREQ porque no puede firmar correctamente su mensaje RREP, sólo se comporte como si

no tuviera la ruta y reenvíe el mensaje RREQ. La segunda es que, cada vez que un nodo genera un mensaje RREQ, él también incluya flags de RREP, el tamaño del prefijo y la firma que pueden ser utilizados (por cualquier nodo intermedio que cree una ruta reversa al creador del RREQ) para contestar a un RREQ que pregunta por el nodo que originó el primer RREQ. Por otra parte, cuando un nodo intermedio genera un mensaje RREP, el tiempo de vida de la ruta ha cambiado del original que tenía. Por lo tanto, el nodo intermedio debe incluir ambos tiempos de vida (el anterior que es necesario para verificar la firma del destino de la ruta) y firmar el nuevo tiempo de vida. De esta manera, la información original de la ruta es firmada por el destino final y el tiempo de vida es firmado por el nodo intermedio.

Para distinguir los diferentes mensajes de la extensión SAODV, los que tienen dos firmas se llaman RREQ y RREP con *Extensión de Doble Firma* (RREQ-DSE y RREP-DSE).

Cuando un nodo recibe un RREQ, primero verifica la firma antes de crear o actualizar una ruta reversa a ese host. Solamente si se verifica la firma, grabará la ruta. Si el RREQ fue recibido con una Extensión de Doble Firma, entonces el nodo también almacenará la firma para el RREP y tiempo de vida (que es valor del 'tiempo de vida de la ruta hacia atrás o reversa') en la entrada de la ruta. El nodo intermedio contestará a un RREQ con un RREP solamente si satisface los requisitos de AODV para hacerlo y el nodo tiene la firma correspondiente y el tiempo de vida anterior para poner en los campos Signature y Old Lifetime del RREP-DSE. De lo contrario, difunde nuevamente (rebroadcast) el RREQ.

Cuando un RREQ es recibido por el destino en sí mismo, contestará con un RREP solamente si satisface los requisitos del AODV para hacerlo. Este RREP será enviado como un RREP con *Extensión Firma Única* (RREP-SSE).

Cuando un nodo recibe un RREP, primero verifica la firma antes de crear o actualizar una ruta a ese host. Solamente si se verifica la firma, grabará la ruta con la firma del RREP y el tiempo de vida.

Usar **firmas digitales** previene los escenarios de **ataque 1 y 3**.

Los mensajes RREP-ACK pueden ser autenticados usando RREP-ACK con Extensión de Firma (RREP-ACK-SE). Además los bloques “Signature Method... Padding” están incluidos antes del campo “Signature” en todos las extensiones de mensajes, y antes del campo “Signature of the new Lifetime and Originator IP address” en el mensaje de RREP-DSE.

SAKM especifica la lista de los valores posibles del campo "Signature Method" y cómo las claves públicas y las firmas son codificadas en la extensión de los mensajes, este tema se explica en la próxima sección.

Mensajes de error en SAODV

Los mensajes de RERRs son protegidos de una forma diferente ya que contienen una gran cantidad de información mutable. Por lo tanto, cada nodo

(que genera o reenvía un mensaje RERR) usa firmas digitales para firmar el mensaje completo y cualquier vecino que reciba el mensaje verifica la firma. Y, puesto que los números de secuencia de destino no son firmados por el nodo correspondiente, un nodo nunca debe actualizar ningún número de secuencia de destino de su tabla de encaminamiento basada en un mensaje de RERR (esto evita que un nodo malicioso realice un **ataque del tipo 6 – Ver sección 2.2.4**).

Cuando un nodo reinicia

El tipo de **ataque 7 (Ver sección 2.2.4)** estaba basado en el hecho de que el creador del RREQ puede fijar el número de secuencia del destino. En el caso que todos se comporten según el protocolo, la situación en la cual el creador de un RREQ pondría un número de secuencia destino más grande que el verdadero nunca sucederá. Sin embargo, esto no ocurre hasta el caso en que destino del RREQ ha reiniciado. Después de reiniciar, el nodo no recuerda más su número de secuencia, sino que espera un período de tiempo suficiente antes de volverse activo, de modo que cuando lo haga nadie haya almacenado su anterior número de secuencia.

Para evitar este ataque, en el caso que el número de secuencia de destino en el RREQ sea más grande que el número de secuencia en el nodo destino, el nodo destino no considerará el valor en el RREQ. En su lugar, se dará cuenta que el creador del RREQ no se está comportando correctamente y enviará el RREP con el número de secuencia correcto. Además, si uno de los nodos tiene una manera de almacenar su número de secuencia cada vez que lo modifica, puede hacerlo. Por lo tanto, cuando reanuda no necesitará esperar tanto tiempo para que todos supriman las rutas hacia él.

3.3.4.4. SIMPLE AD HOC KEY MANAGEMENT(SAKM)

Simple Ad Hoc Key Management es un sistema de administración de claves que permite que los nodos de una red ad hoc utilicen criptografía asimétrica sin depender de un servicio centralizado. Está diseñado para ser aplicado a protocolos de encaminamiento de redes ad hoc que proveen características de seguridad que requieren el uso de criptografía asimétrica (tales como SAODV o SDYMO).

SAKM protege los campos **no-mutables** de los mensajes de encaminamiento.

Debido a que en una red ad hoc no se puede asumir que un nodo participante siempre sea alcanzable por todos los otros nodos, los nodos no pueden cumplir la función de servidores en el sentido convencional de las redes fijas.

Asimismo, la asociación entre claves públicas y otros atributos de un nodo se realiza típicamente usando certificados de clave pública. Un acercamiento podría ser una autoridad de certificación (CA) para publicar tales certificados. Esto es razonable si los nodos ad hoc pudieran tener direcciones

fijas. Sin embargo, el direccionamiento en redes ad hoc es probable que siga las tendencias recientes hacia la asignación de dirección dinámica y autoconfiguración. En estos esquemas un nodo escoge una dirección tentativa y comprueba si ya está en uso mediante un broadcast de una consulta. Si no se encontró ningún conflicto, se permite al nodo utilizar esa dirección. Si se encuentra un conflicto, se requiere que el nodo escoja otra dirección tentativa y repita el proceso.

Entonces, si la dirección IP no identifica a un nodo (dado que son dinámicas y que tampoco existen servidores dedicados como en las redes fijas, como DHCP, por ejemplo), cómo hace un nodo para conocer la dirección IP de un nodo al cual quiere enviar datos o para determinar su propia dirección IP? La solución sería asociar la dirección dinámica recién generada a las claves públicas.

En resumen, lo que se requiere de un sistema considerando las características de las redes ad hoc es que: las direcciones IP sean generadas dinámicamente, los nodos sean identificables mediante sus direcciones IP, que exista una asociación entre la clave pública y la dirección IP de un nodo y que todo esto sea realizado sin ninguna clase de autoridad de certificación.

En el contexto de IP móvil la solución propuesta al problema de la "propiedad de direcciones" consiste en escoger un par de claves, y mapear la clave pública a una dirección tentativa de alguna forma determinista. En este enfoque se usan las llamadas direcciones estadísticamente únicas y criptográficamente verificables (**SUCV, Statistically Unique and Cryptographically Verifiable**). Las direcciones SUCV son generadas mediante el hashing de una "imprint" más la clave pública. Este imprint (que puede ser un valor al azar) se utiliza para limitar ciertos ataques relacionados con IP móvil.

El esquema SAKM genera direcciones IP de una manera similar a la anterior, sin embargo, sólo es necesario realizar el hash de la clave pública. El resumen o digest formateado de alguna forma específica (para que sea una dirección válida) será lo que se conoce como "Dirección Generada Criptográficamente" (**"Cryptographically Generated Address"-CGA**), la cual también será estadísticamente única. Cuando un mensaje usa CGA como dirección IP fuente e incluye la clave pública de un nodo y es firmado por su clave privada, cualquier otro nodo puede verificar que el nodo tiene determinada identidad (representada por el conocimiento de su clave secreta).

3.3.4.5. Generación de direcciones IP para SAODV

En SAODV, se recomienda utilizar IPv6 (en lugar de IPv4) debido a la mayor longitud de la dirección (que garantizaría la unicidad estadística de las direcciones IP). La dirección puede ser, entonces, un prefijo de red de 64 bits con un SAODV_HID de 64 bits (Half Identifier) o un prefijo SAODV_FID de 128 bits (Identifier). Estos dos identificadores se generan casi de la misma

forma que el sucvHID y el sucvID en SUCV (con la diferencia que se aplica hash a la clave pública en vez de aplicarlo a una huella):

SAODV_HID = SHA1HMAC_64 (PublicKey, PublicKey)
SAODV_FID = SHA1HMAC_128 (PublicKey, PublicKey)

Habrá una bandera en las extensiones de los mensajes de encaminamiento SAODV ("H") que se establecerá en '1' si la dirección IP es HID o en '0' si es FID.

3.3.4.6. Nuevos campos en mensajes SAODV

La clave pública se debe incluir en los mensajes de encaminamiento que se firman, de modo que los nodos puedan verificar la firma. Puesto que, obviamente, esa clave pública se debe firmar, se pone antes del campo de firma.

El identificador del algoritmo que se utiliza para firmar el mensaje se especifica en el campo Signature_Method. Los valores posibles se muestran en la tabla 2 (siendo obligatorio soportar RSA). Puesto que SAODV podría permitir más de un método posible de firma, puede ser que suceda que un nodo tenga que verificar una firma con un método que no conoce. Si esto sucede el nodo considerará que la verificación de la firma ha fallado.

Esto implica que todos los nodos que formen parte de una red MANET debe conocer todos los métodos usados por todos los otros nodos para firmar sus mensajes. Sin embargo, no representa un problema puesto que, normalmente, todos los nodos de una red de MANET utilizarán el mismo método (o a lo sumo dos métodos diferentes). El hecho que haya más de un método de firma posible es porque diversas redes pueden tener requisitos de seguridad más estrictos que algunos otras y, por lo tanto, utilizar diferentes métodos de la firma.

Valor	Método de Firma
0	Reservado
1	RSA
2	DSA
3	Curva Elíptica
4-127	Reservado
128-255	Dependiente de la implementación

Tabla 2. Posibles valores del campo Signature Method

3.3.4.7. Detección de IPs duplicadas

Si un nodo 'A' recibe un mensaje de encaminamiento que está firmado por un nodo 'B' que tiene la misma dirección IP que uno de los nodos para los cuales 'A' tiene una entrada de ruta (nodo 'C'), no procesará normalmente ese mensaje de encaminamiento. En su lugar, informará a 'B' que está utilizando

una IP duplicada y lo probará agregando la clave pública de 'C' (así 'B' pueda verificar la veracidad de lo que afirma).

Cuando el nodo 'B' recibe un mensaje de encaminamiento que indica que alguien más esta utilizando la misma dirección IP que él (o se da cuenta por sí mismo), tendrá que generar un nuevo par de claves pública/privada. Después de eso, obtendrá su dirección IP a partir de su clave pública y puede informar a todos los otros nodos (mediante un broadcast) de cuál es su nueva dirección IP con un mensaje especial que contenga: las dos direcciones IP (la anterior y la nueva) y las dos firmas públicas (la anterior y la nueva) firmadas con la anterior clave privada y, todo esto, firmado con la nueva clave privada. Sin embargo, es mucho mejor si ese mensaje es unicast (en lugar de broadcast) a todos los nodos que considere que deban recibir esta información (en el caso de sean sólo unos pocos). Este unicast será contestado con un mensaje de reconocimiento por el receptor si verifica que todo está en orden.

Después de esto, el nodo generará un mensaje RERR para su anterior dirección IP. Su propagación eliminará las entradas de ruta para la anterior dirección IP y, por lo tanto, elimina las direcciones duplicadas. Este mensaje RERR puede tener una extensión del mensaje que diga cuál es la nueva dirección. De esta manera, los nodos que reciben el mensaje de encaminamiento pueden crear ya la ruta a la nueva dirección IP.

SAODV puede ocuparse del problema de direcciones IP duplicadas según lo descrito antes. Un mensaje de detección de dirección duplicada (*Duplicate Address Detected - DADD*) se envía para notificar a un nodo que su dirección ya está siendo utilizada por otro nodo. Un mensaje de notificación de nueva dirección (*New Address - NADD*) se utiliza para informar que el nodo tiene un cambio en su par de claves y en la dirección IP. Finalmente, un mensaje de reconocimiento de nueva dirección (*New Address Acknowledgment - NADD-ACK*) se utiliza para confirmar la recepción del mensaje NADD. En SAODV, el mensaje de NADD es siempre unicast (nunca broadcast).

En esta nueva propuesta, los nodos ad hoc normalmente nunca serán líderes de red. Los líderes de red serán solamente nodos fijos que dan acceso a la red fija y los nodos en la red MANET deberían saber sus direcciones IP, tamaño del prefijo y claves públicas. Los líderes de red no cambiarán su dirección IP en caso de que exista un nodo en la MANET que haya generado la misma dirección IP.

3.3.4.8. Verificación retrasada de firmas

Las firmas de SAODV pueden requerir potencia de procesamiento y que puede ser excesiva para ciertas clases de escenarios ad hoc. La verificación retrasada en la tabla de encaminamiento que están pendientes de verificación, las cuales serán verificadas cuando el nodo tenga tiempo de procesador libre o antes de que estas trata este problema revisando nuevamente uno de los requisitos de la seguridad de SAODV. Los requisitos de seguridad que serán

proporcionados son la autenticación de fuente y la integridad (que combinadas proporcionan la autentificación de datos) y la **importación retrasada de autorización**.

La importación retrasada de autorización permite tener entradas de ruta y eliminaciones de entrada de rutas entradas se deban utilizar para remitir los paquetes de datos.

3.3.4.8.1. Cómo se realiza la Importación Retrasada de Autorización

Cuando un nodo recibe un mensaje de encaminamiento, crea una nueva entrada en su tabla de encaminamiento (la supuesta "ruta reversa"). Por lo tanto, después del broadcast del RREQ, todos los nodos en la red (o en el anillo de difusión) han creado las rutas reversas al creador de la solicitud de ruta. De todas estas rutas reversas, la mayor parte de ellas expirarán pronto (usualmente todas excepto aquellas que estén en la trayectoria seleccionada a través de la cual la respuesta de ruta viajará).

No hay necesidad de verificar todos estos RREQs hasta que el correspondiente RREP retorne y el nodo conozca que está en la trayectoria seleccionada. Las otras rutas reversas expirarán sin ser verificadas.

Realmente, las dos firmas (la del RREQ y la del RREP) serán verificadas después de que el nodo haya reenviado la respuesta de ruta. De esta manera las transmisiones de las solicitudes y de las respuestas de ruta ocurren sin ninguna clase de retraso debido a la verificación de las firmas.

Siguiendo la misma idea, la firma de los mensajes RERR (y generalmente cualquier mensaje de encaminamiento que tenga que ser reenviado) se pueden también verificar después de reenvío de los mismos.

Las rutas pendientes de verificación no serán utilizadas para reenviar ningún paquete. Si un paquete llega para un nodo para el cual haya una ruta pendiente de verificación, el nodo tendrá que verificarlo antes de usar esa ruta. Si la verificación falla, suprimirá la ruta y solicitará una nueva.

3.3.4.9. SAODV con verificación retrasada

Cuando un nodo necesita enviar o reenviar un paquete a un destino para el cual no tenga una ruta activa, primero comprobará si tiene una ruta pendiente de validación. Si lo hace, intentará validarla y, si fue validada con éxito, la marcará como activa y la utilizará. Si después de todo esto no hay una ruta activa el nodo comenzará un proceso de descubrimiento de ruta.

Como se muestra en la figura 4, solamente una vez que la validación se hace con éxito, la ruta es incorporada en la tabla de encaminamiento del nodo. En la tabla de encaminamiento del sistema operativo del nodo: los paquetes se pueden encaminar normalmente, y solamente cuando hay operaciones de búsqueda de ruta que la tabla de encaminamiento no puede resolver, la petición es capturada por el demonio del encaminamiento de SAODV.

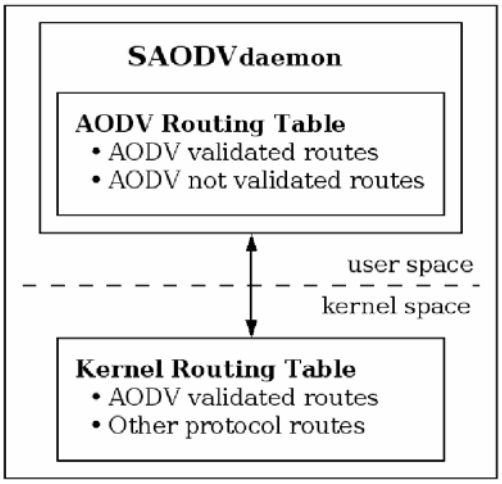


Figura 4. Demonio SAODV

Los nuevos formatos de mensajes se muestran en la sección 3.3.4.9.

3.3.4.10. Formato de paquetes SAODV

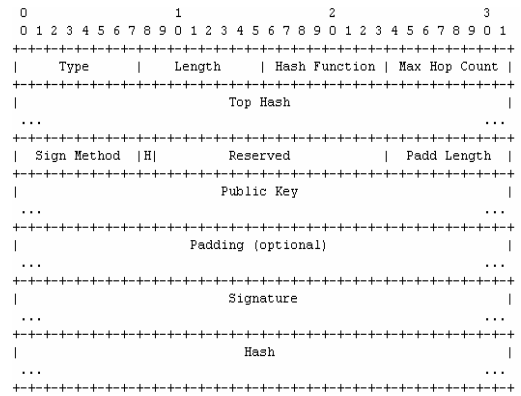


Figura 5 RREQ Single Signature Extension (RREQ-SSE)

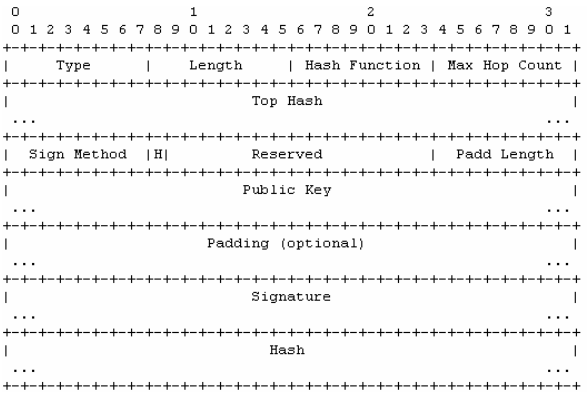


Figura 6 RREP Single Signature Extension (RREP-SSE)

Field	Value
Type	64 in RREQ-SSE and 65 in RREP-SSE
Length	The length of the type-specific data, not including the Type and Length fields of the extension in bytes.
Hash Function	The hash function used to compute the Hash and Top Hash Fields.
Max Hop Count	The Maximum Hop Count supported by the hop count authentication.
Top Hash	The top hash for the hop count authentication. This field has variable length, but it must be 32-bits aligned.
Signature Method	The signature method used to compute the signatures.
H	Half Identifier flag. If it is set to '1' indicates the use of HID and if it is set to '0' the use of FID.
Reserved	Sent as 0; ignored on reception.
Padding Length	Specifies the length of the padding field in 32-bit units. If the padding length field is set to zero, there will be no padding.
Public Key	The public key of the originator of the message. This field has variable length, but it must be 32-bits aligned.
Padding	Random padding. The size of this field is set in the Padding Length field.
Signature	The signature of the all the fields in the AODV packet that are before this field but the Hop Count field. This field has variable length, but it must be 32-bits aligned.
Hash	The hash corresponding to the actual hop count. This field has variable length, but it must be 32-bits aligned.

Tabla 3. Campos de RREQ-SSE y RREP-SSE

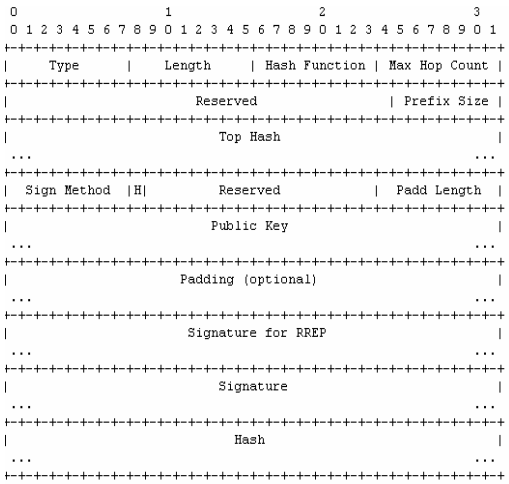


Figura 6. RREQ Double Signature Extension (RREQ-DSE)

Field	Value
Type	66
Length	The length of the type-specific data, not including the Type and Length fields of the extension in bytes.
Hash Function	The hash function used to compute the Hash and Top Hash fields.
Max Hop Count	The Maximum Hop Count supported by the hop count authentication.
Reserved	Sent as 0; ignored on reception.
Prefix Size	The prefix size field for the RREP (it is 7 bit long to allow IPv6 prefixes).
Top Hash	The top hash for the hop count authentication. This field has variable length, but it must be 32-bits aligned.
Signature Method ... Padding	The same than in RREQ (Single) Signature Extension.
Signature for RREP	The signature that should be put into the Signature field of the RREP Double Signature Extension when a intermediate node (that has previously received this RREQ and created a reverse route) wants to generate a RREP for a route to the source of this RREQ. This field has variable length, but it must be 32-bits aligned.
Signature	The signature of the all the fields in the AODV packet that are before this field but the Hop Count field. This field has variable length, but it must be 32-bits aligned. Both signatures are generated by the requesting node.
Hash	The hash corresponding to the actual hop count. This field has variable length, but it must be 32-bits aligned.

Tabla 4. Campos de RREQ-DSE

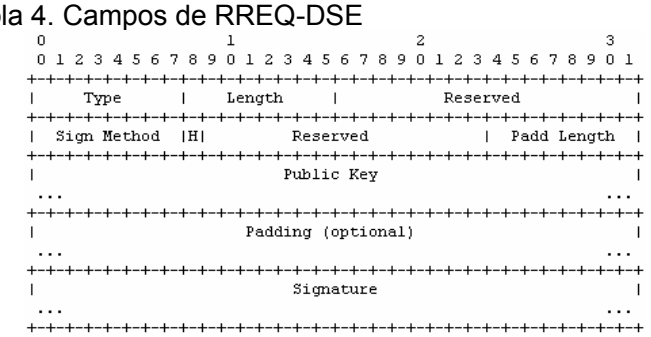


Figura 7. RERR Signature Extension (RERR-SE)

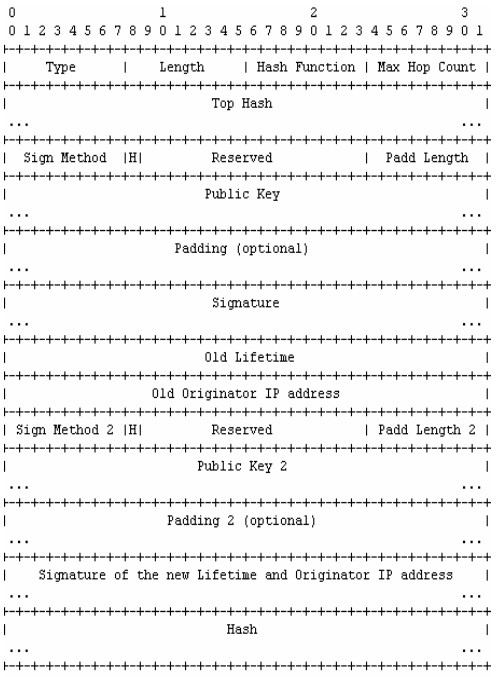


Figura 8. RREP Double Signature Extension (RREP-DSE)

Field	Value
Type	68 in RERR-SE and 69 in RREP-ACK-SE
Length	The length of the type-specific data, not including the Type and Length fields of the extension in bytes.
Reserved	Sent as 0; ignored on reception.
Signature Method ... Padding	The same than in RREQ (Single) Signature Extension.
Signature	The signature of the all the fields in the AODV packet that are before this field. This field has variable length, but it must be 32-bits aligned.

Tabla 5. Campos de RERR y RREP-ACK-SE

Field	Value
Type	67
Length	The length of the type-specific data, not including the Type and Length fields of the extension in bytes.
Hash Function	The hash function used to compute the Hash and Top Hash fields.
Max Hop Count	The Maximum Hop Count supported by the hop count authentication.
Top Hash	The top hash for the hop count authentication. This field has variable length, but it must be 32-bits aligned.
Signature Method ... Padding	The same than in RREQ (Single) Signature Extension.
Signature	The signature of all the fields of the AODV packet that are before this field but the Hop Count field, and with the Old Lifetime value instead of the Lifetime. This signature is the one that was generated by the originator of the RREQ-DSE). This field has variable length, but it must be 32-bits aligned.
Old Lifetime	The lifetime that was in the RREP generated by the originator of the RREQ-DSE).
Old Originator IP address	The Originator IP address that was in the RREP generated by the originator of the RREQ-DSE).
Signature Method 2 ... Padding 2	The whole block of fields is repeated. This time for the 'Signature of the New Lifetime and Originator IP address' signature.
Signature of the new Lifetime and Originator IP address	The signature of the RREP with the actual lifetime (the lifetime of the route in the intermediate node) and with the actual Originator IP address. This signature is generated by the intermediate node. This field has variable length, but it must be 32-bits aligned.
Hash	The hash corresponding to the actual hop count. This field has variable length, but it must be 32-bits aligned.

Tabla 6. Campos de RREP-DSE

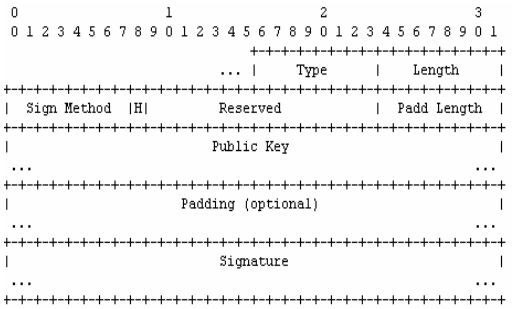


Figura 9. RREP-ACK Signature Extension (RREP-ACK-SE)

3.3.4.11. FORMATO DE MENSAJES DE DETECCIÓN DE IP DUPLICADAS SAODV

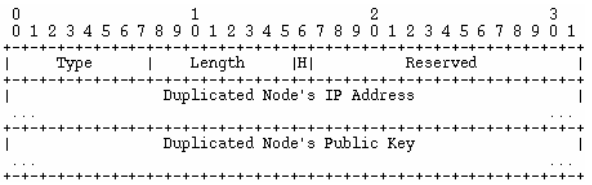


Figura 10. Duplicated Address (DADD) Detected Message

Field	Value
Type	64
Length	The length of the type-specific data, not including the Type and Length fields of the message.
H	Half Identifier flag. If it is set to '1' indicates the use of HID and if it is set to '0' the use of FID.
Reserved	Sent as 0; ignored on reception.
Duplicated Node's IP Address	The IP Address of the node that uses a Duplicated IP Address.
Duplicated Node's Public Key	The Public Key of the node that uses a Duplicated IP Address.

Tabla 7. Campos de mensaje Duplicated Address (DADD) Detected

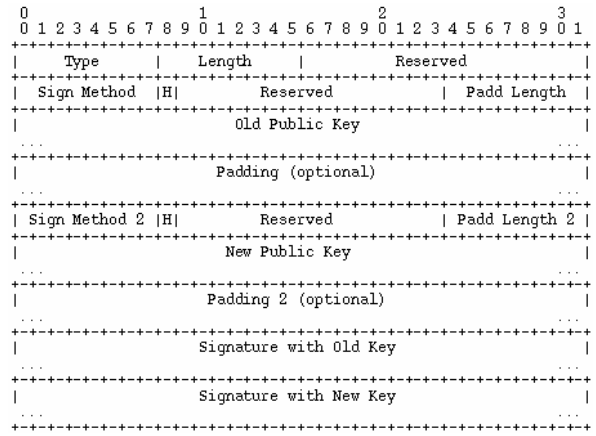


Figura 11. New Address (NADD) Notification Message

Field	Value
Type	65
Length	The length of the type-specific data, not including the Type and Length fields of the message.
Reserved	Sent as 0; ignored on reception.
Signature Method ... Padding	The same than in the Message Extensions. Corresponds to the 'Signature with Old Public Key' signature.
Signature Method 2 ... Padding 2	The whole block of fields is repeated. Corresponds to the 'Signature of the New Public Key' signature.
Signature with Old Key	The signature (with the old key) of the all the fields in the AODV packet that are before this field.
Signature with New Key	The signature (with the new key) of the all the fields in the AODV packet that are before this field.

Tabla 8. Campos de mensaje New Address (NADD) Notification

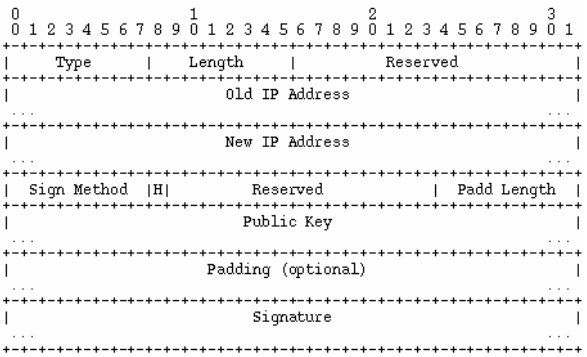


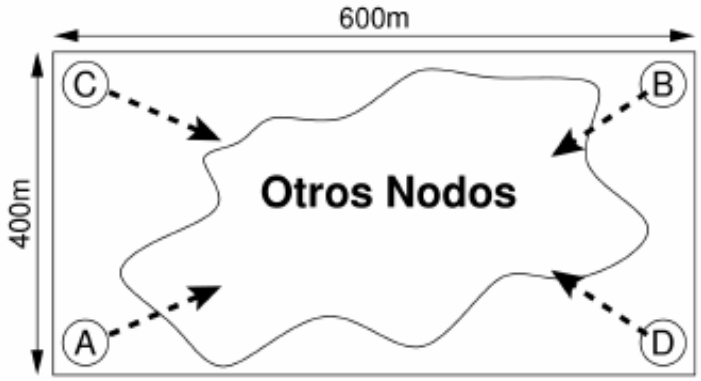
Figura 12. New Address Acknowledgment (NADD-ACK) Message

Field	Value
Type	66
Length	The length of the type-specific data, not including the Type and Length fields of the message.
Reserved	Sent as 0; ignored on reception.
Old IP Address	The old IP address.
New IP Address	The new IP address.
Signature Method ... Padding	The same than in the Message Extensions.
Signature	The signature of the all the fields in the AODV packet that are before this field.

Tabla 9. Campos de mensaje New Address Acknowledgment (NADD-ACK)

3.3.5. Comparativa entre AODV y DSR.

Para llevar a cabo las simulaciones utilizamos Network Simulator 2 (NS2)2 debido a su amplia difusión y aceptación, disponible en el sitio http://www.isi.edu/nsnam/ns/, para el estudio de las MANET. Es bien sabido que los resultados obtenidos con cualquier simulador de red no siempre son comparables con el comportamiento de una red real debido a las simplificaciones de los modelos utilizados. Este inconveniente que presentan los simuladores por software está siendo ampliamente investigado para mejorar la calidad de los resultados y reflejar mejor el comportamiento de una red real. Más allá de esto, Network Simulator es uno de los simuladores más utilizados y los resultados son aceptados por la comunidad internacional de investigadores.



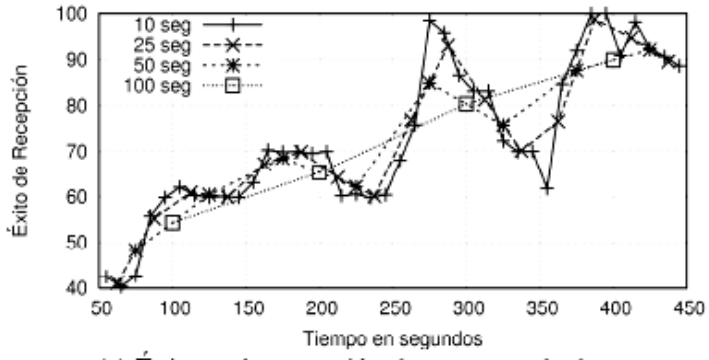
Las simulaciones tienen una duración de 500s y se realizaron en un escenario de 600×400 metros, donde se colocaron 12 nodos, 4 de ellos permanecen estáticos y el resto se desplazan de acuerdo al esquema de movimiento *Random Waypoint (RW)*. En la figura anterior puede verse la posición de los nodos que permanecen estáticos durante toda la simulación (A, B, C y D). La nube representa a los nodos restantes, que toman posiciones iniciales aleatorias y se desplazan con velocidades aleatorias uniformemente distribuídas entre 1 y 4m/s. Los tiempos de pausa para RW varían entre 0s (movimiento constante sin pausa) y 500s (estaticidad). Si bien RW, al igual que otros esquemas de movimiento, nunca alcanza el estado de estaticidad (*steady state*) no consideramos a esto como un inconveniente para el estudio que estamos presentando. Sin embargo, para mejorar el desempeño de RW se puede utilizar velocidad mínima distinta de cero, motivo por el cual establecimos la velocidad en 1m/s.

Para los nodos de la nube se establecieron 10 escenarios por cada tiempo de pausa elegido, obteniéndose un conjunto de 50 escenarios diferentes. En estos escenarios se simularon 4 conexiones unidireccionales de datos con fuentes de tasa constante (*Constant Bit Rate, CBR*) sobre el protocolo de datagramas de usuario (*User Datagram Protocol, UDP*), enviando 15 paquetes por segundo de 512 bytes cada uno. De estos cuatro enlaces de datos, dos se establecieron para el par de nodos A-B (desde A hacia B y desde

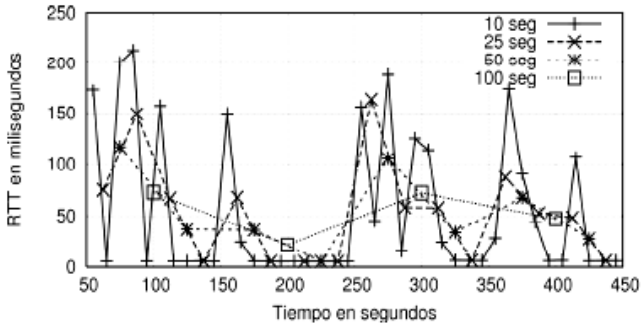
B hacia A) y los dos restantes para el par C-D (desde C hacia D y desde D hacia C), siendo equivalente a utilizar 2 enlaces bidireccionales.

A nivel de enrutamiento se utilizaron los protocolos de enrutamiento AODV y DSR. A nivel MAC se utilizó la capa IEEE 802.11 configurada para velocidades de datos de 11Mbps. Las simulaciones se realizaron con y sin la utilización del esquema de portadora virtual RTS/CTS (*Request to Send/Clear to Send)*, que tiene como objetivo la reserva del canal y que además ayuda en la solución del problema del *nodo oculto*. Se utilizó *Two Ray Ground* como esquema de propagación de las señales de radio y no se utilizó ganancia en las antenas.

En resumen, se simularon y promediaron 4 enlaces de datos entre dos pares de nodos y se establecieron 10 escenarios por cada uno de los 5 tiempos de pausa aplicados a los nodos restantes. En todos los escenarios se utilizaron los protocolos AODV y DSR con y sin la utilización RTS/CTS de IEEE 802.11. Del total de 200 simulaciones realizadas, solamente presentamos los resultados obtenidos sin la utilización del mecanismo RTS/CTS debido a la similitud existente con los resultados en donde sí se utilizó dicho mecanismo.



(a) Éxito en la recepción de paquetes de datos



(b) Tiempo medio de viaje de los paquetes de datos

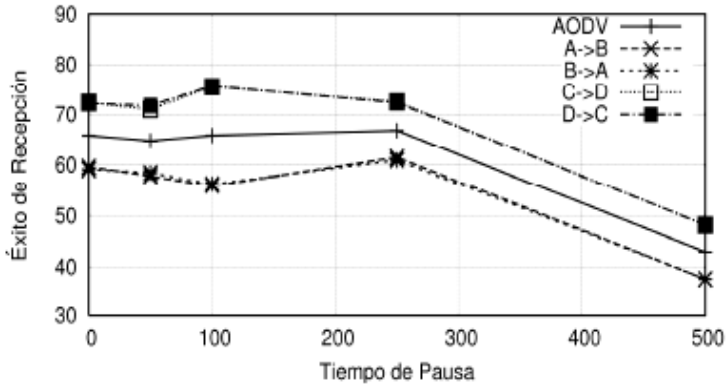
3.3.5.1. Análisis de los resultados.

Para llevar a cabo esta nueva forma de estudiar el desempeño de la red, primero aplicamos diferentes tamaños de ventana de tiempo a un enlace en un conjunto de simulaciones y así poder elegir el tamaño apropiado para el tipo de dato que se vaya a analizar. En la última figura se pueden ver las diferencias existentes entre las ventanas de tiempo escogidas para una misma configuración de simulación; cada punto corresponde al centro de la ventana de tiempo. Para mostrar esto, escogimos las datos obtenidos para el enlace desde el nodo C hacia el nodo D (C→D), simulando la red con tiempo de pausa de 250s, sin RTS/CTS y con el protocolo AODV. Nótese como se enmascaran las variaciones experimentadas por el enlace a medida que se incrementa el tamaño de la ventana de tiempo utilizada. En la gráfica superior se puede ver la evolución del éxito en la recepción de paquetes de datos y en la inferior el tiempo medio de viaje (*Round Trip Time, RTT*) de los paquetes recibidos. Como puede verse en las figuras, la utilización de una ventana de tiempo grande (100s) oculta información y una ventana de tiempo pequeña (10s) puede presentar datos muy confusos. Para los datos obtenidos en las simulaciones realizadas para este artículo, la ventana de tiempo de 25s demostró ser apropiada ya que presenta suficiente precisión y suaviza las variaciones de alta frecuencia.

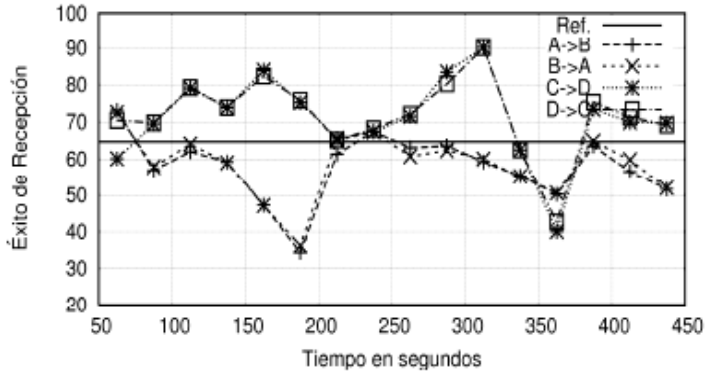
Los gráficos que presentamos a continuación reflejan las diferencias encontradas al cambiar la forma de estudiar la red y se contrastan con lo utilizado actualmente. Para evitar los posibles estados transitorios de inicio y fin de la simulación, hemos descartado los 50s iniciales y finales para el cálculo de todos los datos que presentamos en este trabajo. Cabe destacar que los escenarios y el nivel de tráfico utilizados favorecen al desempeño general de la red, pero el objetivo de este trabajo es presentar otra forma de analizar los resultados obtenidos y no el estudio y comparación del comportamiento de los protocolos de enrutamiento.

3.3.5.2. Éxito de recepción.

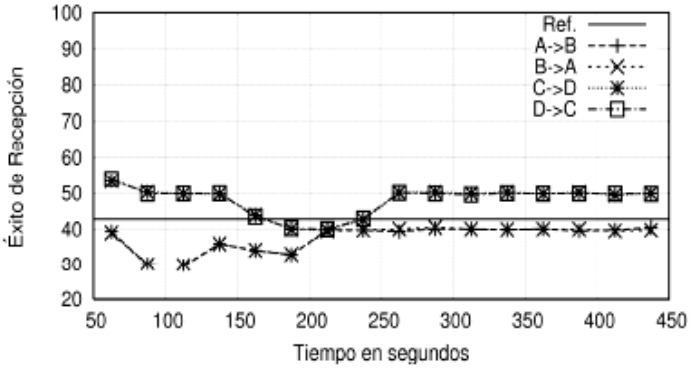
Una de las mediciones más utilizadas es la de *Éxito de recepción* de paquetes de datos ya que refleja como se comportó la red con respecto al establecimiento, utilización y mantenimiento de los enlaces. En la Fig. 3(a) se puede ver el éxito de recepción de paquetes de datos versus el tiempo de pausa utilizado en los diferentes escenarios para el protocolo de enrutamiento AODV. La tendencia decreciente de las curvas está relacionada con la probabilidad de realizar transmisiones exitosas. Tal como fue presentado en, comprobamos que la movilidad incrementa la probabilidad de éxito ya que se renueva con mayor frecuencia la cantidad de enlaces posibles, hecho que no ocurre cuando los nodos se encuentran estáticos. En la figura que se muestra a continuación se puede apreciar la diferencia existente entre los caminos utilizados para cada par de nodos transmisor/receptor. La polarización en dos grupos de curvas se corresponde con el par de nodos entre los que se establece el enlace de datos (los nodos A y B por un lado y los nodos C y D por el otro). Cabe destacar que entre los pares de nodos existe una diferencia de hasta un 20% que no se refleja en la curva del promedio general.



(a) Éxito en recepción promedio y por enlace (origen→destino).



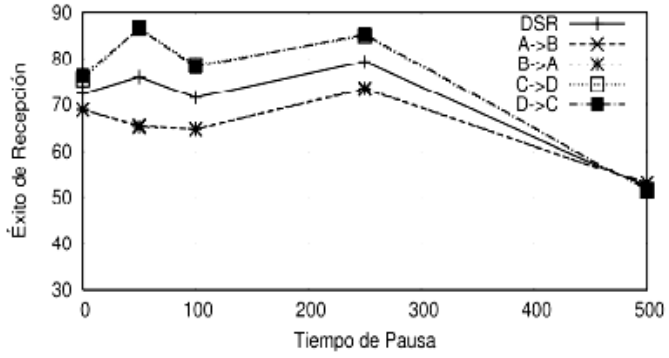
(b) Éxito por ventanas de tiempo para pausa de 50 segundos.



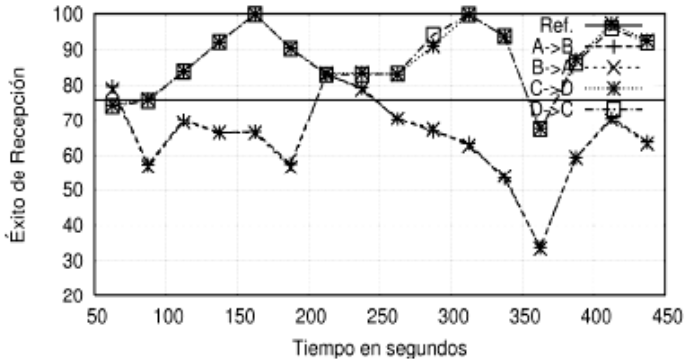
(c) Éxito por ventana de tiempo para pausa de 500 segundos.

Tomando como ejemplo, sin perder generalidad, los resultados de las simulaciones para tiempo de pausa 50 y 500s se obtuvieron las Fig. b c. La

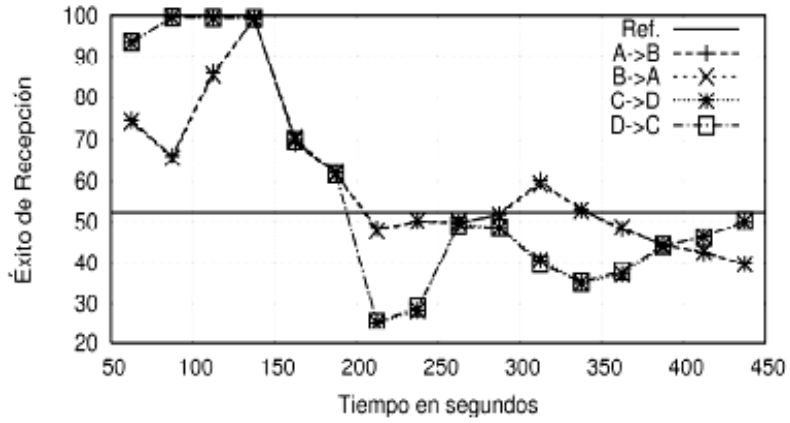
recta de referencia es el valor del promedio de todos los enlaces, que se corresponde con lo presentado en la Fig. a, y el resto de las curvas corresponden a los promedios por ventana de tiempo de 25spara cada enlace. Al discriminar por enlace y a lo largo de la simulación se puede observar que la variabilidad de éxito de recepción experimentada por cada par de nodos es notable, alcanzando diferencias de hasta un 50% (Fig. 3(b)). Cabe destacar la notable diferencia existente entre el desempeño del par A-B, que se encuentra por debajo de la línea de referencia, y el desempeño del par C-D, cuya operación está por encima la línea de referencia.



(a) Éxito en recepción promedio y por enlace (origen→destino).



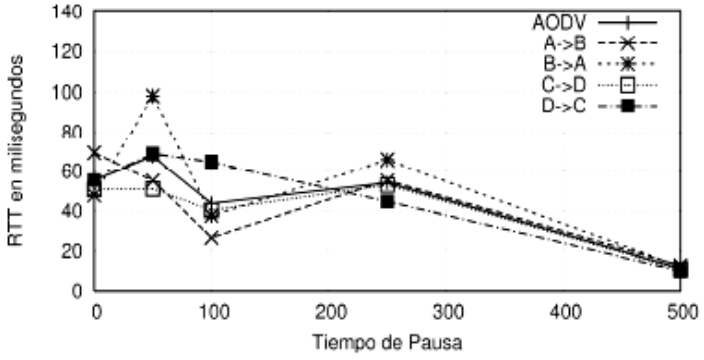
(b) Éxito por ventana de tiempo para pausa de 50 segundos.



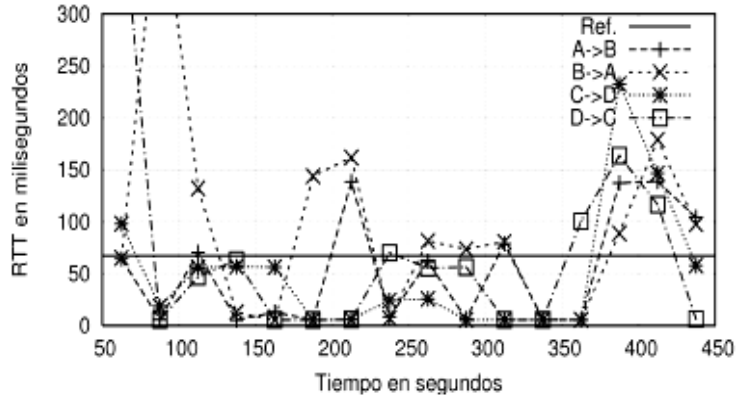
(c) Éxito por ventana de tiempo para pausa de 500 segundos.

Puede verse en la Fig. a que el desempeño del protocolo de enrutamiento DSR con respecto al éxito en la recepción de los paquetes de datos es superior al observado con el protocolo AODV. También se observa un mejor desempeño para el par de nodos C-D. En este caso, la diferencia entre las dos rutas utilizadas llega a superar el 20% pero disminuye a medida que disminuye la movilidad de los nodos.

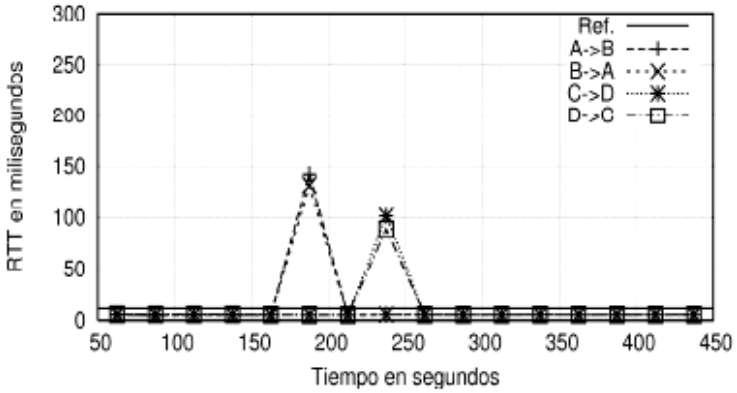
En la Fig. b se observa un comportamiento similar al visto en la anterior figura para tiempo de pausa de 50 s, donde la variabilidad máxima alcanza una diferencia de poco más del 30%. Por el contrario, los resultados reflejados en la Fig. c difieren bastante de los observados en la Fig. c debido a que el éxito en la recepción disminuye con el paso del tiempo. En la primer cuarta parte del tiempo la operación es muy buena, pero en el resto del tiempo de simulación decae a valores no apropiados. Inclusive, la variabilidad que sufren los enlaces supera el 70%, problema que se oculta completamente en los datos obtenidos del promedio general.



(a) RTT general (AODV) y por enlace (origen→destino).



(b) RTT por ventana de tiempo para pausa de 50 segundos.

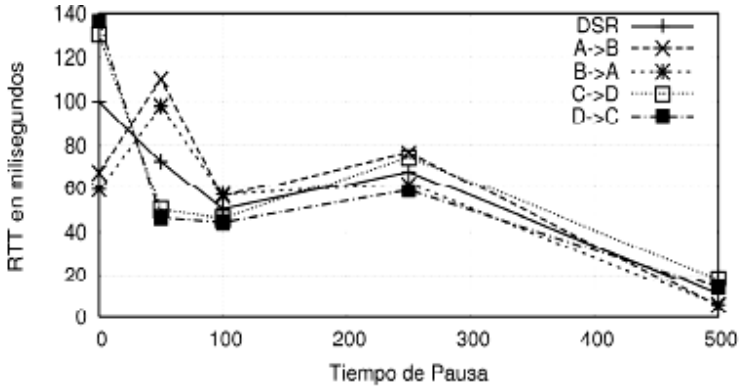


(c) RTT por ventana de tiempo para pausa de 500 segundos.

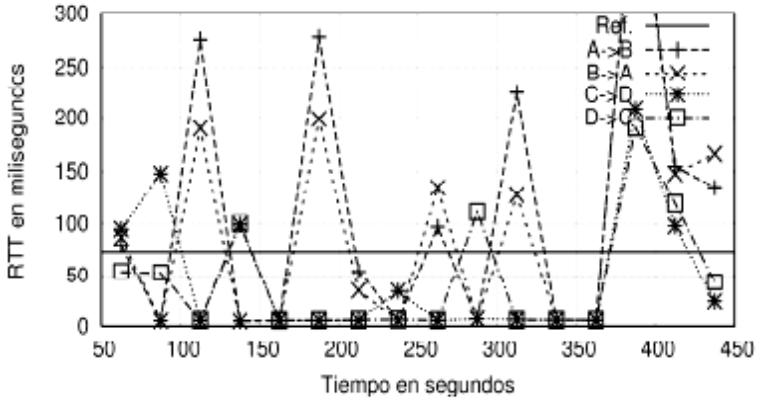
La discriminación de la información por enlace muestra que el comportamiento ante la evolución de la red es diferente.

3.3.5.3. Tiempo Medio de Viaje

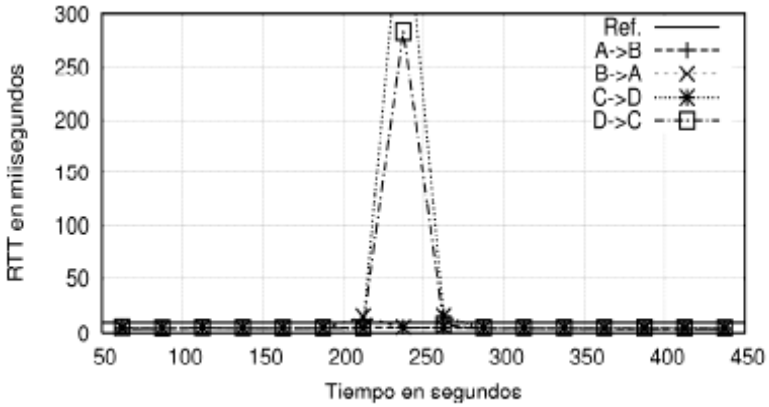
El tiempo medio de viaje (*RTT*) brinda información valiosa cuando se desea verificar que los enlaces de datos sean aptos para la transmisión de datos sensibles en tiempo. En general, un servicio interactivo de audio, video o ambos medios combinados se considera de calidad aceptable cuando los usuarios experimentan retardos no mayores a los 75ms. Si se considera el promedio general, puede verse en la Fig. a que el protocolo AODV pudo desempeñarse correctamente ante esta restricción para todos los escenarios utilizados. Sin embargo, si se observan los desempeños particulares de cada enlace, algunos de ellos son muy buenos y otros no tanto, existiendo entre ellos diferencias de hasta 50ms.



(a) RTT general (DSR) y por enlace (origen→destino).



(b) RTT por ventana de tiempo para pausa de 50 segundos



(c) RTT por ventana de tiempo para pausa de 500 segundos.

Nuevamente, consideramos los resultados para los casos de 50 y 500s de pausa para analizar los resultados discriminados por enlace y en ventanas de tiempo. En la Fig. b se observa una variabilidad muy alta que puede superar los 300ms, haciendo que el promedio se encuentre la mayor parte del tiempo por encima del retardo experimentado en cada enlace. Para los resultados con

tiempo de pausa de 500s (Fig. c), es notable la estabilidad que presentan las curvas producto de la estaticidad de los nodos pertenecientes a la nube. Si bien en este caso el promedio general es más acorde al comportamiento de todos los enlaces, este se encuentra por encima de los valores por enlace y se pierden los picos transitorios debidos a la caída de un enlace. En el caso del protocolo DSR, puede verse en la Fig. a que, en líneas generales, tanto el promedio general y los valores por enlace tienen un comportamiento similar. La mayor diferencia existe para alta movilidad en los nodos (tiempo de pausa pequeño) donde la diferencia con el promedio general puede superar los 30ms y la diferencia entre enlaces los 70ms. Sin embargo, los resultados observados son aceptables para la transmisión de datos sensibles en tiempo. En la Fig. b puede verse que el promedio general se encuentra dentro de los límites para considerar aceptable la calidad del enlace en cuanto a RTT. Sin embargo, el promedio general oculta el alto retardo experimentado por los datos que intercambian el par de nodos A-B. Los retardos para este enlace, especialmente para la conexión desde A hacia B, están en general por encima de los 100ms con una variabilidad de más de 300ms. Para el caso de tiempo de pausa de 500s (Fig. c), al igual que ocurre con AODV, el promedio general es coherente con la mayoría de los datos, pero se observa un pico de más de 200ms. La diferencia entre la Fig. c y la Fig. c está en los picos de la zona de los 200s, en la segunda aparece solo uno de los picos debido a la forma de trabajo del protocolo de enrutamiento. Recordemos que DSR mantiene información de caminos alternativos (si existen) que toma del tráfico de red, mecanismo que le permite encontrar rutas más cortas y mejor reacción ante la caída de un enlace.

3.3.5.4. Conclusiones

En este capítulo se ha mostrado que la utilización de los gráficos clásicos para el estudio de las MANET oculta comportamientos propios de este tipo de redes que deben ser considerados cuando se estudia enrutamiento con provisión de calidad de servicio para servicios interactivos de tiempo real. La movilidad, la aparición/desaparición de nodos, el medio donde se propaga la señal, entre otros factores, hacen a estas redes de datos completamente variables en el tiempo. Por lo tanto, la observación de algún parámetro mediante promedios generales no siempre es apropiada para entender cómo evoluciona la red. La utilización de los gráficos que se proponen tiene como objetivo principal mostrar aquellos datos que se pierden al promediar datos en el análisis y estudio de las MANET.

Se han realizado simulaciones sobre un ambiente controlado y se procesaron los resultados obtenidos discriminando por enlace de datos. Este análisis permite distinguir diferencias notables que pueden existir entre diferentes caminos de datos y ver cómo se enmascara esa información al observar solamente los promedios generales. La utilización de ventanas de tiempo por enlace de datos para el estudio del comportamiento de los protocolos permite obtener información más detallada de la evolución de las conexiones. Este tipo de gráficos deja en evidencia la evolución temporal de los enlaces y ofrece información valiosa para considerar a la hora del desarrollo de protocolos de enrutamiento. Los resultados presentados muestran que en las

MANET solo puede brindarse calidad de servicio en algunos momentos y bajo ciertas condiciones (algunos autores utilizan el término *Soft-QoS* para establecer la diferencia con la calidad de servicio de las redes estáticas).

4. Estudio y comparación entre AODV y SAODV

4.1. Ventajas e inconvenientes del uso de SAODV frente a AODV

SAODV tiene inconvenientes cuando es comparado con AODV en cuanto al promedio de retraso mayor en la comunicación y el mayor consumo de potencia por los nodos (debido al cálculo de las primitivas de la criptografía asimétrica: la firma y verificación del mensaje), para lo cual se propone como solución la verificación retrasada de autorización.

Se ha realizado simulaciones hechas por el creador de SAODV en una red con 30 nodos moviéndose a una velocidad máxima de 10m/seg en una superficie de 1000x1000 mts, usando los protocolos AODV, SAODV con RSA, SAODV con ECC (Criptografía de Curvas Elípticas) y SAODV2 (SAODV con verificación retrasada) con ECC utilizando longitudes de clave de seguridad equivalentes (1368 bits para RSA y DSA y 160 bits para ECC). Se han simulado pruebas para medir diferentes parámetros, tales como el retraso en la transmisión de paquetes extremo-a-extremo, la tasa de entrega de paquetes y la carga de encaminamiento normalizada (normalized routing load). Para las dos últimas pruebas no hubo prácticamente diferencia entre los resultados (90% de tasa de entrega de paquetes y carga de encaminamiento normalizada de 1). Como resultado se puede ver en la figura 13 que el retraso en la entrega de paquetes entre AODV y SAODV2 ECC son similares, es decir que es la mejor de las opciones con respecto a las otras versiones de SAODV.

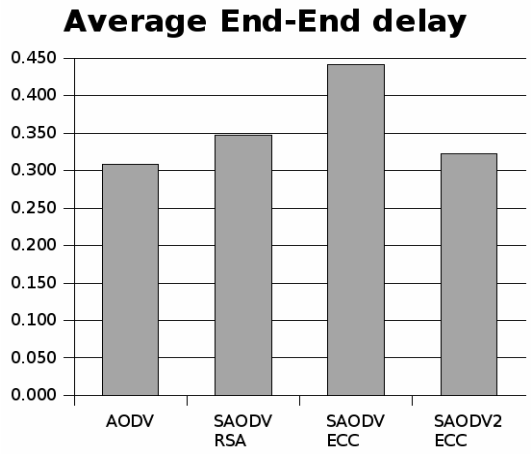


Figura 13. Resultados de simulación.

En cuanto a los formatos originales de los paquetes AODV debemos decir que se han modificado para agregar extensiones a los paquetes RREQ, RREP, RERR y RREP-ACK. Las extensiones consisten en campos que se utilizan para implementar las firmas digitales y cadenas hash. Por otra parte se agregaron nuevos paquetes tales como RREQ y RREP con Extensión de Doble Firma (RREQ-DSE y RREP-DSE) y los paquetes DADD, NADD, NADD-ACK para la generación de direcciones IP CGA.

La firma digital *Digital_signature $_X$ (routing message)* solamente puede ser creada por *X*. Así, sirve como prueba de validez de la información contenida en el mensaje de encaminamiento. Esto previene los escenarios de **ataque 1, 3, 4, y 6**, descritos en la sección 2.2.3.

El autenticador de salto reduce la capacidad de que un nodo intermedio malicioso monte un ataque de **tipo 2** arbitrariamente modificando el contador de salto sin detección. Un nodo que está a *n* saltos lejos de *T* conocerá el e*nésimo* elemento en la cadena hash ($h^n(x)$), pero no conocerá ningún elemento que se encuentre antes de éste debido a la propiedad de unidireccionalidad de la función *h()*. Sin embargo, el nodo malicioso podría todavía transmitir el autenticador recibido y el contador de saltos sin modificarlos. Así, la eficacia de este acercamiento es limitada.

Asimismo es necesario mencionar que hay otro tipo de ataque que no puede detectar SAODV: *ataques mediante túneles*. En este tipo de ataque, dos nodos maliciosos simulan que tienen un enlace entre ellos (es decir, pueden enviar y recibir mensajes directamente el uno al otro). Alcanzan esto enviando mensajes AODV a través de un túnel entre ellos (probablemente de una manera cifrada). De esta forma podrían llegar a tener cierto tráfico a través de ellos. Al parecer, no hay esquema de seguridad, hasta ahora, capaz de detectar esto. Los esquemas de detección de comportamiento incorrecto podrían, en principio, detectar los supuestos ataques de túneles. Si el monitor ve un mensaje de encaminamiento con *Hop_Count = X+1* que está siendo enviado por un nodo pero no vió un mensaje de encaminamiento con *Hop_Count = X* siendo enviado al mismo nodo, entonces el nodo o está inventando el mensaje de encaminamiento o existe un túnel. En cualquier caso es razón para accionar la alarma. Sin embargo, esta clase de esquema tiene como principal problema que no hay manera para que todos los nodos validen la autenticidad de los informes del comportamiento incorrecto y existe la posibilidad de detectar falsamente nodos que no se comporten correctamente. Por lo tanto, no es una solución factible hasta ahora.

Por otra parte la utilización de los números de secuencias debe prevenir la mayor parte de los posibles *ataques de respuesta*. Un nodo desechará un mensaje respondido si ha recibido un mensaje original ya que el mensaje contestado no será “lo suficientemente fresco”. Para hacer más robusta la prevención de ataques de respuesta, un nodo podría considerar incrementar su número de secuencia en más situaciones que en las que es obligatorio para AODV(o aún periódicamente).

Podría ser posible montar un ataque malicioso modificando la cabecera IP de los mensajes SAODV. Sin embargo, en la opinión del creador de SAODV esto no es posible ya que SAODV no confía en el contenido de la cabecera lP, y toda la información que necesita para funcionar está dentro del mensaje de AODV y de la extensión de SAODV.

Se muestran las pruebas experimentales reales realizadas por investigadores de la Universidad de Columbia Británica en Canadá y de Samsung Electronics de Korea para comparar el rendimiento de SAODV con AODV. Utilizando una plataforma experimental de pequeña escala, consistente en 10 equipos móviles dentro de un campo de rugby de 250x100mts, se realizaron tres tipos de pruebas: simulación con ns-2, emulación interior usando MacSim, y pruebas en el exterior con equipos reales.

Para cada prueba se seleccionaron al azar 3 fuentes y destinos entre los 10 nodos y los flujos de información o sesiones que se establecieron por cada vez entre cada par de nodos fueron tanto UDP como TCP.

La implementación del protocolo AODV usada fue AODV-UU(de la Universidad de Uppsala- Suecia).

La implementación de SAODV fue una modificación de AODV-UU, usando MD5 como función hash y parte del código de la implementación de ARAN (Authenticated Routing for Ad-Hoc Networks), el cual utiliza la librería OpenSSL para la certificación. Se deshabilitó la posibilidad de que los nodos intermedios puedan firmar RREP, es decir que sólo el destino pudo firmar el mensaje RREP.

Con respecto a los ataques se realizó una modificación del código de AODV-UU para que cuando un atacante reciba un RREQ responda con un RREP con campo Hop-Count=0. Sólo se consideraron las pruebas con un sólo atacante.

Para tráfico UDP se definieron la **tasa de entrega de paquetes** = cociente entre el número de paquetes de datos a destino y el número de paquetes generados por todas las fuentes de tráfico, y la **sobrecarga de control** = (RREQ+ RREP+ RERR) generados durante cada ejecución de las pruebas. Para tráfico TCP se consideró el **promedio de throughput**.

En general se ha comprobado que SAODV es efectivo en la prevención de los ataques por falsificación de mensajes de encaminamiento y por eliminación de paquetes de datos. A continuación se mencionan los resultados.

4.2. Resultados de simulación y emulación indoor

(1) *Tráfico UDP*

- *Tasa de entrega de paquetes UDP*. Sin atacantes AODV y SAODV se comportan de las mismas forma, más del 90% de entrega de paquetes. SAODV entrega un mayor porcentaje de paquetes en presencia de un atacante tanto en la simulación como en la emulación interior.

- *Sobrecarga de Control de encaminamiento (en paquetes)*, es decir, la cantidad de paquetes de enrutamiento recolectados durante la ejecución de la prueba. Cuando no hay atacantes son similares los valores. Existe

más sobrecarga en el flujo de paquetes RREQ que en los otros porque se vuelven a difundir en la red cuando se hace el descubrimiento de ruta. Sin embargo en presencia de un atacante se ve como los paquetes RREP se incrementan con respecto a los RREP que se envían cuando no hay atacantes, ya que el atacante falsifica RREP para cualquier RREQ recibido. SAODV presenta una sobrecarga de control de encaminamiento en presencia de un atacante.

- *Sobrecarga de control de encaminamiento (en bytes)*. En SAODV siempre es mayor debido a los campos adicionales de los paquetes.

(2) *Tráfico TCP*

- *Promedio de throughput TCP*. Sin atacantes la sesión 3 tiene un mayor rendimiento que las sesiones 1 y 2 bajo la simulación interior, esto se debió a que estas últimas tenían caminos a 2 saltos y compartían un nodo intermedio más frecuentemente que la sesión 3 (representaban cuello de botella). Se concluye que existe una diferencia entre el tráfico UDP y TCP en el caso de los ataques por falsificación de paquetes de control y eliminación de paquetes de datos. Para el tráfico UDP cualquier paquete eliminado por un atacante puede nunca ser recuperado. Sin embargo, el ataque por eliminación de paquetes puede no ser tan efectivo en los flujos TCP especialmente en un entorno móvil cuando un atacante no es capaz de mantenerse el mismo en una ubicación para ser un nodo intermedio por un período largo de tiempo.

- *Throughput TCP en AODV* en la emulación interior con y sin atacantes. Se identifican momentos cuando un atacante tiene éxito y bloquea el flujo de tráfico, sin embargo, durante estos instantes, otros flujos TCP se benefician con el rendimiento más alto debido a la menor carga de tráfico en la red.

- *Sobrecarga de Control de encaminamiento en paquetes*. El valor más alto corresponde a RREQ con respecto a RREP y RERR. También hay un mayor porcentaje en los paquetes RREP cuando hay ataques.

- *Sobrecarga de control de encaminamiento en bytes*, nuevamente SAODV tiene el valor más alto, debido a los campos adicionales en los paquetes RREQs y RREPs.

4.3. Resultados en pruebas exteriores

(1) *Comunicación en zona gris*

La llamada zona gris de comunicación es una situación que ocurre en AODV cuando el nodo destino recibe correctamente paquetes de control pero no paquetes de datos. Las pruebas demostraron que el ancho de la zona gris de comunicación es alrededor de 9m y 18m bajo la tasa de transmisión de 2Mb/s y 11Mb/s respectivamente. Cuando un nodo se encuentra en una zona

gris, el enlace puede no ser declarado roto (por ejemplo, los paquetes HELLO puede aún ser recibidos ocasionalmente, los RERR pueden no ser enviados) aunque se pierdan muchos paquetes.

(2) *Tráfico UDP*

- *Tasa de entrega de paquetes*: cuando no hay atacantes, muestran una alta tasa de entrega tanto AODV como SAODV. Cuando existe un atacante, SAODV tiene una mayor tasa de entrega de paquetes para las 3 sesiones.

- *Sobrecarga de Control de encaminamiento en paquetes*. SAODV no introduce un incremento significativo en la cantidad de paquetes de control transmitidos. Sin embargo, debido al tamaño de paquete mayor para SAODV, la sobrecarga agregada en el encaminamiento es mayor para SAODV que para AODV. A pesar de eso, los resultados de las pruebas de la tasa de entrega de paquetes muestran que SAODV es efectivo en la prevención de los ataques por falsificación de mensajes de control y eliminación de mensajes de datos bajo tráfico UDP.

(3) *Tráfico TCP*

- *Promedio de throughput TCP* para cada sesión. Sin atacante se muestra un alto rendimiento TCP. Cuando hay un atacante las 3 sesiones decrecen en un 50% su rendimiento, sin embargo SAODV tiene el mayor promedio.

- *Sobrecarga de control de encaminamiento (en paquetes)*. A pesar de que SAODV tiene una mayor sobrecarga de control debido a los campos adicionales para llevar la información de certificados, el resultado muestra que la sobrecarga de control extra no disminuye significativamente el rendimiento TCP. Estos resultados muestran que SAODV es efectivo en la prevención de los ataques por falsificación de mensajes de control y eliminación de mensajes de datos bajo tráfico TCP.

4.4. Conclusiones

La seguridad en redes ad hoc inalámbricas en el ámbito de protocolos de encaminamiento es un aspecto muy importante a ser considerado durante el diseño de estos protocolos, debido a las amenazas de seguridad que presentan dichas redes.

Sin embargo también hay que considerar los escenarios en los cuales operará la red ad hoc. AODV es adecuado en aquellos escenarios donde se asume que todos los nodos son amigables y que confían los unos en los otros, por lo tanto la seguridad no es una característica indispensable ni necesaria.

Por otra parte SAODV es preciso en aquellos escenarios donde se forma una red ad hoc de manera libre y por consiguiente no existe confianza

entre los nodos. Este protocolo, por tanto, no sería apto para escenarios militares donde se requiere el control de nodos comprometidos, tampoco cuando se requiere disponibilidad permanente.

Es importante mencionar que, como en todos los protocolos, SAODV no soluciona por completo el problema de ataques por DoS (ninguno puede prevenir estos ataques si se dan en la capa física). SAODV se utilizará entonces, cuando las características de seguridad que se deben cumplir como mínimo sean la integridad y la autenticidad de los mensajes de encaminamiento y autenticidad de fuente, es decir de los nodos que envían tales mensajes.

Asimismo, un protocolo de encaminamiento seguro debe contar con un mecanismo de seguridad que satisfaga sus requerimientos, en el caso de SAODV está representado por la aplicación de las cadenas hash a los mensajes de encaminamiento y la utilización de firma digitales mediante el sistema de administración de claves SAKM.

Con SAODV no hay manera de impedir que un nodo invente muchas identidades, que no puedan ser utilizadas para crear un ataque contra el algoritmo encaminamiento seguro. Sin embargo, un atacante no puede suplantar a otro nodo, y un nodo puede siempre probar que es el mismo nodo.

La verificación retrasada hace posible que un nodo malicioso cree solicitudes de ruta inválidas que podrían inundar la red MANET. Pero, el mismo nodo malicioso puede inundar la red con solicitudes de ruta perfectamente válidas. Y no habría manera posible de saber si está intentando inundar la red o si sólo está intentando ver si algunos de sus nodos amigos están presentes en la red, por ejemplo.

En SAODV un atacante no puede falsificar un par claves pública/privada a partir de una dirección IP, así el símbolo de identidad de un participante de la red se convierte en la dirección IP en sí misma.

SAODV con verificación retrasada y ECC obtiene un rendimiento mejor que las otras combinaciones de SAODV y un rendimiento muy cercano a la implementación de AODV, por lo tanto podría esperarse que esta versión de SAODV llegue a ser utilizado para dar seguridad a redes ad hoc que precisan un nivel de seguridad, como ya mencionamos, con integridad y autenticidad de datos así como de los nodos que se comunican, teniendo en cuenta el equilibrio entre seguridad y eficacia.

Finalmente, podemos concluir que dotar de seguridad al protocolo de encaminamiento utilizado en cada nodo de una red ad hoc depende de los escenarios donde opere la misma, sin embargo la utilización de SAODV puede ser beneficiosa desde el punto de vista de implementar seguridad aunque se requiera un mayor procesamiento por parte de cada nodo, pero que es insignificante con los beneficios de poder asegurar la comunicación de los nodos de la red hoc.

5. Conclusiones

5.1. Conclusiones

Las redes MANET están en auge, es indudable. Tras 9 meses dedicados prácticamente de forma exclusiva al estudio de este tipo de redes hemos podido comprender la dificultad de diseñar un protocolo eficiente. A lo largo de la memoria hemos observado como mejorar un aspecto del protocolo implicaba empeorar otro, de forma indirecta e inevitable. Por tanto es muy necesario alcanzar un compromiso entre todas las características y aspectos deseados en éstas redes.

Tras estudiar varios protocolos hemos visto como el AODV desde nuestro humilde punto de vista parecía cumplir de mejor manera todos los compromisos que hemos mencionado. Al ser una mejora del DSR en la que se reduce el tamaño de las cabeceras de los paquetes se mejora el ancho de banda. Al ser un protocolo reactivo se reducen los paquetes de de control, reduciendo así las colisiones y mejorando de nuevo el ancho de banda efectivo respecto a los protocolos proactivos. Por desgracia, no se puede garantizar que éste protocolo sea el mejor en todas y cada una de las redes Ad Hoc, ya que para ciertas características será mejor escoger un protocolo u otro. Pero creemos que, generalmente, en caso de desconocimiento del comportamiento de los nodos de la red y del número de nodos que la formen, el protocolo AODV será el más acertado.

Además, hemos sido capaces de estudiar como sería una posible modificación de este protocolo, el SAODV, el cuál añade un sistema de seguridad basado en claves *hashing* y firmas digitales. Este añadido no implica un empeoramiento significativo del ancho de banda y, por el contrario, si implica grandes ventajas ya que, por desgracia, actualmente todo tráfico de datos es necesario que este dotado de cierta seguridad. Como ya hemos comentado a lo largo de la memoria, nodos maliciosos, egoístas pueden perjudicar el funcionamiento de la red o “escuchar” datos confidenciales. La seguridad en las redes inalámbricas es una materia pendiente por muchos protocolos, pero que de aquí a un futuro cercano dejará de serlo para convertirse un aspecto inherente a todo protocolo.

5.2. Debilidades

5.2.1. Debilidades en redes ad hoc.

Los principales aspectos en los que se investiga actualmente en el ámbito de las redes Ad-Hoc son:

- La escalabilidad, tal vez el principal problema. Hoy con apenas 50 o 100 nodos los resultados empiezan a ser insatisfactorios.

- El ahorro de energía. La batería es un bien escaso y muy preciado, es muy importante buscar formas de optimizar su aprovechamiento. Hay técnicas como dormir nodos para despertarlos en determinados

intervalos de tiempo, lo que exige una sincronización no trivial. Algunos sistemas permiten transmitir con más o menos fuerza, modificando el alcance de la transmisión.

- La premisa de la buena fe: las redes Ad-Hoc se basan en la cooperación bienintencionada entre estaciones. En las redes ordinarias hay que tratar los problemas ocasionados por nodos maliciosos, pero en estas redes la cuestión es aún más compleja porque no es necesaria la hostilidad para causar daño, basta la ausencia de altruismo. Por ejemplo, si el nivel de la batería está bajo puede ser legítimo no gastarlo retransmitiendo para los demás, o limitarse a los mensajes más urgentes. Pero esto puede dar lugar a abusos.

- El modelo cliente-servidor es el habitual en Internet, pero no es adecuado en redes Ad-Hoc, ya no hay entidades bien conocidas que ofrezcan servicios. Precisamente una pregunta relevante es ¿dónde están los servicios?, lo que resulta difícil en entornos tan cambiantes. Los servicios pueden buscarse al mismo tiempo que las rutas, pero esto va contra el modelo de capas, que cuya bondad está demostrada. Una alternativa es basarse en direcciones multicast bien conocidas, esto parece adecuado para servicios básicos como DNS o DHCP, aunque en redes Ad-Hoc es un tema abierto.

- La seguridad: lo que está en el aire puede ser capturado fácilmente. La solución es cifrar las comunicaciones, pero la seguridad y el cifrado se basan en una distribución segura de claves y en una autoridad certificadora centralizada, y esto último es casi una contradicción en términos tratándose de redes Ad-Hoc. Además, el ancho de banda y la capacidad de procesamiento requerido para hacer seguras las comunicaciones pueden suponer una carga muy importante. El multicast (multidifusión), que puede ahorrar drásticamente ancho de banda en distribución masiva. No está claro si debe incluirse en este nivel de los algoritmos. Por un lado, los problemas de multidifusión pueden ser de naturaleza lo bastante específica como para que no merezca la pena mezclarlos con otras cuestiones. Pero por otra parte, los protocolos dedican mucho esfuerzo a conocer el estado de nodos intermedios que cambian continuamente. Si el multicast se convierte en una acumulación de unicast, muy posiblemente se estarán desperdiciando recursos.

- La simetría en los enlaces: si la estación A puede transmitir a la estación B, lo más habitual es que B pueda transmitir a la estación A. Pero en algunas tecnologías concretas no sucede esto, lo que complica notablemente todos los algoritmos.

- La gran variedad de medios físicos distintos, todos incompatibles entre sí. Cuando todos los nodos emplean la misma tecnología en el nivel físico, pueden trabajar juntos, pero solo en ese caso. Esto es un serio inconveniente para llegar a un protocolo estándar.

- La ubicación en la torre de protocolos. Normalmente se hace en nivel de red. Esto presenta circunstancia favorables: en las tablas de encaminamiento se trabaja con direcciones de red que las aplicaciones resuelven en direcciones de enlace: en la dirección de enlace del propio nodo destino si es accesible directamente, o bien la dirección de enlace del nodo que hace de router y que se lo hará llegar. Si la movilidad está en el nivel de enlace, entre otras cosas hay que resolver direcciones de enlace con otras direcciones de enlace, lo que resulta poco natural.

- La calidad de servicio: podemos indicar si los enlaces son válidos o no, pero apenas hay expresividad para indicar cambios en su calidad.

- La adaptación de las aplicaciones al ancho de banda disponible. Usando enlaces inalámbricos, el ancho de banda es típicamente un orden de magnitud inferior al cable. Sería deseable que las aplicaciones fueran conscientes de ello, y en cada caso adapten la información enviada: tal vez reduciendo el número de cuadros por segundo si se trata de vídeo, el número de colores o la resolución en las imágenes, u omitiendo en páginas web animaciones de utilidad dudosa.

- Las máquinas de recursos limitados: En informática, robótica y comunicaciones los equipos continuamente incrementan sus prestaciones y decrementan su precio. Pero siempre habrá dispositivos sencillos capaces de comunicarse con otros sin cables. Esta sencillez puede deberse a diferentes causas: se les puede exigir ser extremadamente fiables, imponer muchas restricciones a su precio (incluso tanto como para hacerlos desechables), pueden tener muy limitada su ocupación del espacio radioeléctrico, pueden ser material docente, juguetes, colonias numerosas de microrobots o nanorobots, etc. Puede que necesiten una vida útil muy larga con serias limitaciones de consumo energético que les obliguen a efectuar solo las comunicaciones imprescindibles. Puede tratarse de PDAs, electrodomésticos, juguetes, equipos industriales, sensores y un largo etcétera. Es muy probable que en el futuro encontremos aplicaciones para este tipo de máquinas que ahora no conocemos. Por todo ello afirmamos que no importa cuánto mejoren los dispositivos, siempre habrá aplicaciones donde las máquinas de recursos limitados sean de utilidad. Si intentásemos llevar el protocolo DSR con direcciones IPv4 a una máquina con características como las que mencionamos o similares, una buena parte del datagrama estaría ocupado por las cabeceras, ya que emplea encaminamiento en origen, cada datagrama incluye la dirección de cada nodo que debe recorrer. La longitud de las cabeceras es variable: tomando como referencia la implementación de DSR disponible para el simulador de redes ns-2 serían necesarios 88 bytes: un tercio del total disponible. Si DSR se usa bajo IPv6 se requerirían 288 bytes, lo que resultaría inviable con la arquitectura que proponemos como ejemplo.

5.2.2. Debilidades de los protocolos estudiados.

A lo largo de toda la memoria hemos tratado, entre otras cosas, de sintetizar las características de distintos protocolos. Para la elección del estudio de estos protocolos, como ya hemos dicho nos hemos basado en las distintas clasificaciones y hemos escogido los más representativos de los distintos grupos.

En el caso del OLSR hemos visto que es un protocolo excelente en cuanto a tiempo de respuesta, ya que al mantener información de la topología de la red sabe en el momento de enviar un paquete la ruta exacta que debe de seguir el paquete. Ahora bien, mantener toda la información de la red requiere una serie de requisitos bastante exigentes que pueden perjudicar el funcionamiento de la misma. Para que un nodo reúna la información pertinente es necesario el continuo envío de paquetes de control entre los diferentes nodos de la red. Estos paquetes han de mandarse continuamente, ya que la movilidad de los nodos, puede obligar a variar la información almacenada previamente. Ahora bien, si el tiempo entre envío de estos paquetes es muy elevado la perspectiva real de la topología de la red se puede perder. Por el contrario, si el tiempo es excesivamente pequeño, la visión topológica es perfecta a costa de saturar el ancho de banda de las comunicaciones de la red, aumentando las probabilidades de colisión.

Aparte, es necesario que cada nodo almacene y procese la información retransmitida en los paquetes. Por tanto, es necesario una estructura compleja y un tamaño de memoria elevado y con un tiempo de acceso muy corto para evitar latencias innecesarias. Con lo cual, el precio de cada nodo puede ser bastante elevado.

Como ya se puede intuir, el uso de este protocolo en una red MANET sería muy adecuado en topologías muy cambiantes y a ser posible, con un número no muy elevado de nodos, de esta forma, el número de paquetes de control que circulan en un instante determinado de tiempo es pequeño, aprovechando mejor el ancho de banda efectivo. A la par, al ser una red “pequeña” la memoria no tiene porque ser muy grande, lo que abarata el coste de la misma.

Para los protocolos reactivos estudiados, bien el DSR, el AODV o el SAODV, las características se invierten. Es decir, donde el OLSR ofrecía un rendimiento excelente, a saber, el tiempo de latencia de reenvío de paquetes, en estos protocolos este tiempo es bastante más elevado, debido a las etapas de descubrimiento de ruta necesarias para el envío de paquetes.

De la misma forma se invierte los requisitos de uso. Es decir, gracias a un correcto uso de las caches de almacenamiento de rutas se puede usar un protocolo reactivo en redes estáticas, o al menos con escasa movilidad, alcanzando un rendimiento excelente. Esto es debido a que la etapa del descubrimiento de la ruta solo haría falta realizarla en el primer envío de un

paquete. En los envíos sucesivos se usará la información almacenada en la memoria cache, reduciendo los paquetes de control y el tiempo de latencia.

Por tanto, podemos sintetizar que los protocolos reactivos no obtienen buenos rendimientos en redes de alta movilidad, y que es muy importante optimizar la política de sustitución (o invalidación) de la memoria de cache de rutas.

5.3. Trabajos futuros

Tras haber realizado este estudio durante el último año, creemos conveniente seguir trabjando en el desarrollo de este tipo de protocolos y aportar nuevas ideas. Concretamente destacaríamos los siguientes puntos como objetivos principales:

- Escalabilidad.
- QoS.
- Seguridad.

A día de hoy, ya existen multitud de implementaciones de diferentes protocolos, con enfoques muy dispares, que ofrecen rendimientos más que satisfactorios en redes normalmente reducidas, incluso pudiónse adaptar a frecuentes cambios de topología. Sin embargo, estos protocolos se muestran ineficaces en cuanto ell número de nodos de la red se empieza a ser importante. Esto nos lleva directamente al segundo punto. Algunos parámetros de calidad de servicio, no son del todo controlabes por los protocolos, p.e. la pérdida de paquetes, que están causados por la propia tecnología, por lo el margen de maniobra de nuestro posible protocolo es muy limitado. Sin embargo, existen otros aspectos como el ancho de banda, el tiempo de servicio… que son mejorables y sobre los que habrá que trabajar.

Finalmente habrá que tratar la seguridad, puesto, que una vez que se ha conseguido obtener protocolos, que, por rendimiento, ya pueden ser aplicables en algunos campos, tendremos que tener en cuenta dicho aspecto. Ahora un buen protocolo no será sólo un protocolo eficiente. Si no que deberá también seguro. Se han planteado varias opciones durante este estudio. Sin embargo, hay que tener en cuenta que la calidad de estas alternativas es, como mínimo, directamente proporcional a la pérdida de prestaciones de un protocolo cuando se aplican. Esto nosd obligará a refinar dichos sistemas de seguridad, además de siempre tener presente hasta que límite podemos llegar al dotar de mecanismos de seguridad a nuestro protocolo, si no queremos perder una calidad de servicio demasiado grande. Estos temas serán un reto para nosotros y muchas otras personas, que se hayan inmersas en la investigación de este tipo de redes. Y los años serán los que nos enseñen hasta donde podemos llegar.

ANEXO A.

En la actualidad existen un gran número de simuladores de redes. Sin embargo, a pesar de las mejoras introducidas en los últimos años en estos programas, sus prestaciones todavían son limitadas. No siempre sus resultados se corresponden con los de simulaciones reales, lo cual se debe a las simplificaciones que presentan los modelos empleados por dichos programas. Dentro de este grupo de programas, uno de los que ofrece resultados más completos es el Network Simulator 2, cuyo funcionamiento detallaremos a continuación con un sencillo ejemplo.

Network Simulation 2.

Ns-2 es un simulador de redes, que está escrito en C++. La simulación se escribe en lenguaje Otcl, que es un lenguage orientado a objetos de tipo interprete,es decir, que las instrucciones del código se va traduciendo una a una conforme se va ejecutandose, dandole una flexibilidad durante el desarrollo del código para hacer la simulación.

De esta forma los enlaces entre los nodos (links), son objetos OTcl que influyen y que se pueden programar situaciones como:

- Retrasos
- Gestión de colas
- Módulos de pérdidas
- Errores
- Etc.

Si se quiere modificar alguno de estos párametros o incluir uno propio, emplearemos el C++.

El resto de las funciones se implementan en estos leguajes:

- El enrrutado está casi todo implementado en OTcl.
- Los algoritmos de Dijkra están implementados en C++.

Ns puede trabaja con la shell de OTcl, que es por donde se introducen individualmente los distintos comandos, este es un método lento y complicado.

Para ello utilizaremos el procedimiento de los programas, donde

escribiremos todo el código de la simulación a realizar y despues lo ejecutaremos.

Como no se tiene editor propio utilizaremos el bloc de notas de Windows o cualquier otro que tengamos a mano.

El procedimiento que vamos a utilizar para aprender a manejar el entorno del NS será interactivo, no será necesario saber OTcl o C++, sino que iremos avanzando con los distintos tipos de redes e introduciendo los nuevos conceptos e instrucciones progresivamente.

Una vez hecha la simulación podemos realizar dos cosas para conocer los resultados:
-Ver el funcionamiento de la red simulada en forma de gráfico animado con un programa llamado NAM.
-Tambien podemos analizar la carga de la red a traves de un programa que da a conocer de forma gráfica la relación tiempo carga de la red a traves de XGRAF.

Para cada uno de ellos conforme se realizen las simulaciones los iremos aplicando.

Para más información ir a la página http://www.isi.edu/nsnam/ns/

Instalación sobre Windows.

Los pasos a seguir serán los siguientes

1 Descargar TCL (versión 8.3.5) disponible en http://ftp.activestate.com/ActiveTcl/Windows/8.3.5/ActiveTcl8.3.5.0-2-win32-ix86.exe

2 Instalar TCL

3 Reiniciar el ordenador

4 Crear le directorio para guardar NS, NAM y los ejecutables a simular. C:\ns

5 Descargar el fichero nam-1.0alla-win32.exe en el directorio ns, se puede descargar de http://www.isi.edu/nsnam/dist/binary/nam-1.0a11a-win32.exe

6 Para mayor comodidad lo renombraremos como nam.exe

7 Descargar el fichero ns2-2.1b9a-win32.exe en el directorio ns, se puede descargar de http://www.isi.edu/nsnam/dist/binary/ns-2.1b9a-win32.exe

8 Para mayor comodidad lo renombraremos como ns.exe

Funcionamiento.

Siempre empezaremos un programa de la siguiente forma

set ns [new Simulator]

Esta orden crea una instancia del objeto simulador, diciendole que vamos a realizar una nueva simlación.

Despues abriremos el fichero donde escribiremos los datos obtenidos por el simulador, para despues leerlo con NAM y los represente de forma gráfica.

set nf [pen out.nam w]
$ns namtrace-al $nf

Lo que hacemos es abrir o crear un fichero llamado "out.nam", donde escribiremos y leeeremos los datos creados por la simulación.
En la línea sigiente, los datos creados por NS se guardan en el out.nam que los visualizará.

Debemos poner en el programa un procedimiento "finish", que deberá ser definido al principio del programa, el cual cerrará el fichero de valores de trazado (out.nam) y pondrá en marcha a NAM, este deberá tener la siguiente estructura:

```
poc finish{} {
  global ns nf
  $ns flush-trace
  close $nf
exec nam out.nam &
exit=
}
```

Las siguiente líneas que escribiremos será el tiempo que vamos a simular la red

$ns at<tiempo><elemento>

Donde <tiempo> será el valor en segundos y <elemento> será en que procedimiento se cierra la simulación, pe:

$ns at 5.0 "finish"

Así le decimos a NS que ejecute la simulación durante 5.0 seg y despues ejecute el procedimiento "finish"

En la ultima línea del programa será para que aranque la simulación de la siguiente forma

$ns run

Como en todo programa desearemos poner comentarios, para poderlo entender en lecturas posteriores , estos los haremos así:

#Comentario

Este script no es para ejecutar con NS ya que debermos hacerlo con la red completa, aquí no están definidos ni los nodos, ni los enlaces,etc., que necesitamos en una red y que lo harmos posteriormente.

Dos nodos, una conexión.

La primera simulación real que vamos a realizar es unir dos nodos por una línea duplex tal como aparecen en la imagen.

Estos dos nodos están conectados pur una línea duplex de 1Mbps y la transmisión tarda en ir de uno a otro 10ms, es decir, la distancia que se encuentra uno del otro es de 10ms por la velocidad de propagación de la señal en el medio en que se realiza dicha transmisión. P.e. si es un uan fibra donde v=0.7*c la distancia seria L=0.01*0.7*3 exp 8=2100Km..

Ahora vamos a ir explicando lo que deberemos añadir al script antrior para realizar la simulación

Los nodos en NS se definen con la instrucción

set <nombre nodo>[$ns node]

El comando **set** en OTcl significa que creamos una instancia del objeto a simular. Siempre que cremos nodos debermos poner el comado *"$ns node".*

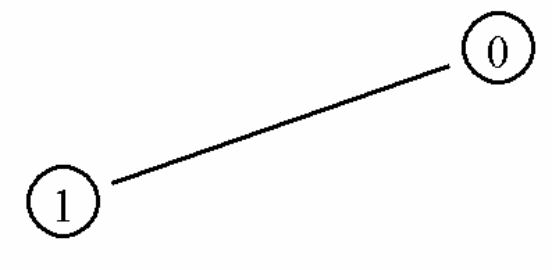
Tenemos dos nodos, n0 y n1, los definiremos en el script de la siguiente forma

ns n0[$ns node]
ns n1[$ns node]

El siguiente paso es unir ambos nodos por una línea, en este caso una línea duplex de la forma

$ns duplex-link $n0 $n1 1Mb 10ms Droptail

El ancho de banda de la línea es de 1Mbit por segundo, que se encuentran ambos nodos a una distancia de 10ms. Saldá un dibujo de la forma siguiente




```
# Creamos el objeto a simular
set ns [new Simulator]
#Abrimos el fichero de trazado NAM
set nf [open out.nam w]
$ns namtrace-all $nf
# Definimos el procedimiento "finish"
  proc finish{} {
       global ns nf
       $ns flush-trace
# Cerramos el fichero de trazado
       close $nf
# Ejecutamos NAM con el fichero de trazado
    exec nam out.nam &
    exit 0
    }
#Definimos los nodos que necesita la simulación
set n0[$ns node]
set n1[$ns node]
# Definimos los enlaces entre nodos
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
#Llamamos a la plicacion "finish" despues de 5segundos de simulación
$ns at 5.0 "finish"
#Arrancamos la simulación planteada
$ns run
```

Enviar datos

El ejemplo anterior solamente vemos su topología, pero en una red debemos enviar datos del nodo n0 al nodo n1. En ns los datos se envian desde un agente a otro. Por tanto el priemr paso será crear el agente que envia los datos del nodo 0 (n0), al agente que recibirá los datos en el nodo 1 (n1).

#Crear un agente UDP y unirlo al nodo n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

Despues deberemos crear un genrador de tráfico, CBR que se unirá al agente UDP. Debemos definir el tamaño de los paquetes a enviar, en este caso los haremos de 500bytes y cada paquete será enviado cada 0.005 segundos, es decir se enviarán 200 paquetes por segundo. Todo esto lo escribiremos así.

#creamos una fuente de tráfico CBR? que se une a udp0
set cbr0 [new Aplication/Traffic/CBR]
$cbr0 set paquetSize_500
$cbr0 set interval_0.005
$cbr attach-agent $udp0

Despues debemos crear el agente que asociaremos all nodo 1, en este caso crearemos un agente nulo.

#Creamos un agente de tipo NULL asociados al nodo 1
set null0 [new Agent/Null]
$ns attach-agent $ns1 $null0

Ahora debemos conectar ambos agentes

#Conectamos el tráfico de la fuente con el del receptor
$ns connect $udp0 $null0

Tambien debemos decir cuando el agente CBR cuando enviará datos y cuando dejará de enviarlos y deberemos hacerlo antes de la línea 'ns at 5.0 "finish" '.

#Fijamos el funcionamiento de eventos de CBR
#Arrancará a los 0.5 seg de empezar la simulación
$ns at 0.5 "$cbr0 start"
#Y hará la parada a los 4.5 segundos
$ns at 4.5 "$cbr0 stop"

Este último código hace que cuando pulsemos el botón de arranque de NAM despues de 0.5 empezará a enviar paquetes del nodo0 al nodo 1. El NAM te

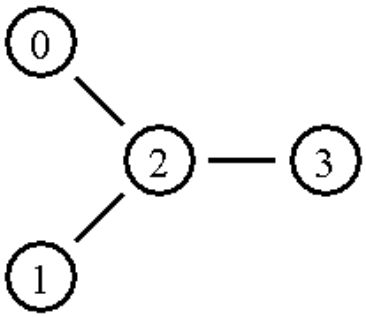
dará un resultado parecido al del dibujo.

```
# Creamos el objeto a simular
set ns [new Simulator]
#Abrimos el fichero de trazado NAM
set nf [pen out.nam w]
$ns namtrace-al $nf
# Definimos el procedimiento "finish"
  poc finish{} {
       global ns nf
       $ns flush-trace
# Cerramos el fichero de trazado
       close $nf
# Ejecutamos NAM con el fichero de trazado
    exec nam out.nam &
    exit 0
    }
#Definimos los nodos que necesita la simulación
set n0[$ns node]
set n1[$ns node]
# Definimos los enlaces entre nodos
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
#Crear un agente UDP y unirlo al nodo n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
#creamos una fuente de tráfico CBR? que se une a udp0
set cbr0 [new Aplication/Traffic/CBR]
$cbr0 set paquetSize_500
$cbr0 set interval_0.005
$cbr attach-agent $udp0
#Creamos un agente de tipo NULL asociados al nodo 1
 set null0 [new Agent/Null]
$ns attach-agent $ns1 $null0
#Conectamos el tráfico de la fuente con el del receptor
 $ns connect $udp0 $null0
#Fijamos el funcionamiento de eventos de CBR
#Arrancará a los 0.5 seg de empezar la simulación
 $ns at 0.5 "$cbr0 start"
#Y hará la parada a los 4.5 segundos
 $ns at 4.5 "$cbr0 stop"
#Llamamos a la plicacion "finish" despues de 5segundos de simulación
$ns at 5.0 "finish"
#Arrancamos la simulación planteada
$ns run
```

Un paso más

En este nuevo apartado vamos a complicar la topología con cuatro nodos donde un nodo enrrutará al resto de los nodos, el resto de los nodos se comunicaran a traves de este.

La estructura a realizar será como la del dibujo



La topología.

Lo primero que debemos definir es la topología que va a tener. Para ell crearemos el fichero "ejemplo2.tcl" usando el código del ejemplo 4 ya que esas partes del código será identicas. Las partes comunes serán:

- Crear el objeto simulador
- Empezaremos la simulación con el mismo comando
- Hacer funcionar automáticamente el NAM abriendo el fichero de trazado, inicializandolo y definiendo el procedimiento de cierre y arranque del NAM

Para crear los cuatro nodos haremos los sigientes pasos

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
```

Despues crearemos las tres líneas duplex que une los nódos con el nodo que hace de router (n2).

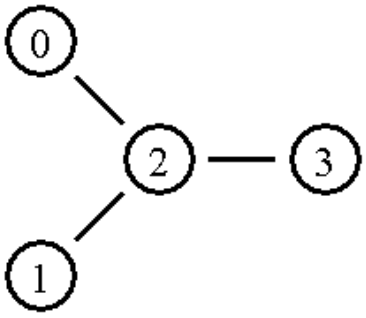
```
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms DropTail
```

Si NAM no dibuja de forma correcta, pulsando el boton "re-layout" para que la dibuje como queremos. Ahora vamos a escribir los canales de conexión entre los nodos y el router

```
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
```

$ns duplex-link-op $n2 $n3 orient right

En estas líneas le decimos a NS que la conexión del nodo 0 al nodo 2 será de izquierda-abajo el de n1 con n2 de izquierda-arriba y del n2 al n3 (ojo) izquierda solamente, quedará como en el dibujo siguiente.



Los eventos.

Ahora crearemos dos agentes UDP con fuentes de tráfico CBR queunimos los nodos n0 y n1 y crearemos un agente nulo que unimos al n3

#Creamos un agente UDP y lo unimos al nodo n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

#Creamos una fuente de trafico CBR que se une a udp0
set cbr0 [new Aplication/Traffic/CBR]
$cbr0 set packetSice_500
$cbr0 set interval_0.005
$cbr0 attach-agent $udp0

#Creamos un agente UDP y lo unimos al nodo n1
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1

#Creamos una fuente de trafico CBR que se une a udp1
set cbr1 [new Aplication/Traffic/CBR]
$cbr0 set packetSice_500
$cbr0 set interval_0.005
$cbr0 attach-agent $udp1

#Creamos un agente nulo para en nodo n3
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0

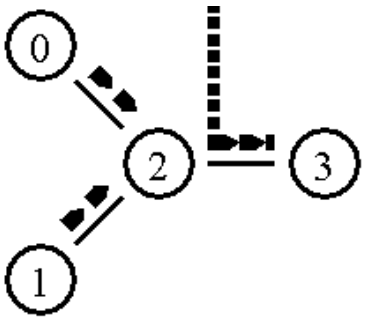
Los dos agentes CBR deberán ser conectados con el agente Null

$ns connect $udp0 $null0
$ns connect $udp1 $null0

El primer CBR arrancará a los 0.5 segundos y parará alos 4.5 seg. Más tarde y el segundo CBR arrancará a 1.0 seg. y parará 4seg. Después

$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"

Cuando se analize el flujo de datos, se verá que hay mas tráfico del n0 con n2 y n1 con n2, que el que lleva la línea de n2 con n3.
El envio de n0 y n1 a n2 es de 200 paquetes de 500bytes, lo que supone utilizar un ancho de banda cada uno de 0.8Mbps pero del n2 al n3 si sumamos el envio de n0 y de n1 hace que entre ambos se encesite un ancho de banda de 1.6 Mbps, cuando hemos definido que tiene un ancho de banda de 1Mbps, por tanto, un porcentaje elevedo de paquetes deberán ser rechazados. Como ambos serán del mismo color no distinguiremos cual es de cada nodo emisor si no pinchamos sobre el paquete en NAM, en los apartados siguientes veremos como se hace esto.



```
#Creamos el objeto a simulador
set ns [new Simulator]

#Abrimos el archivo de trazado
set nf [open out.nam w]
$ns namtrace-all $nf

#Defimos el procedimiento 'finish'
proc finish {} {
	global ns nf
	$ns flush-trace
		#Cerramos el archivo de trazado
	close $nf
		#Execumos en NAM el archivo de trazado
	exec nam out.nam &
	exit 0
}

#Creamos los cuatro nodos
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Creamos la union de los distintos nodos
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms DropTail

#Le indicamos donde queremos que aparezcan en NAM los nodos
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

#Monitor de cola para la union entre el nodo 2 y el nodo 3
$ns duplex-link-op $n2 $n3 queuePos 0.5

#Creamos el agente UDP asociado al nodo n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

#Creamos un tr¨yfico CBR de la fuente asociado a udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#Creamos el agente UDP asociado con el nodo n1
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1

#Crear un trafico CBR asociado a la fuante de udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

#crear un agente nulo (un trafico sink?) asociado al nodo n3
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
#Conectar las fuentes  de trafico con el alsink
$ns connect $udp0 $null0
$ns connect $udp1 $null0
#Planificamos la secuencia de sucesos de los agentes CBR
$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"
#Llamamos al procedimiento finish despues de 5 segundos de simulacion
$ns at 5.0 "finish"
#Hacemos correr la simulacion
$ns run
```

Supervisando el flujo.

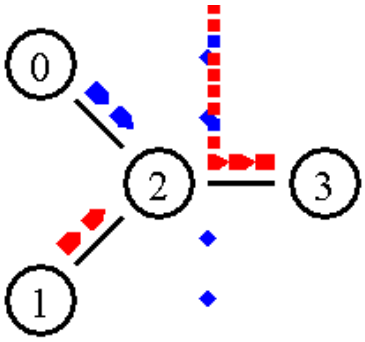
Ahora lo que vamos hacer es añadir dos líneas a las definiciones del agente CBR

$udp0 class_1
$udp1 class_2

Ahora escribamos el siguiente trozo de código y lo insertamos en el script Tcl preferiblemente despues que se haya credo el objeto simulador,puesto que esta es una parte del estado del simulador

$ns color 1 Blue
$ns color 1 Red

De esta forma identificaremos el flujo generado por cada elemento con un color distinto



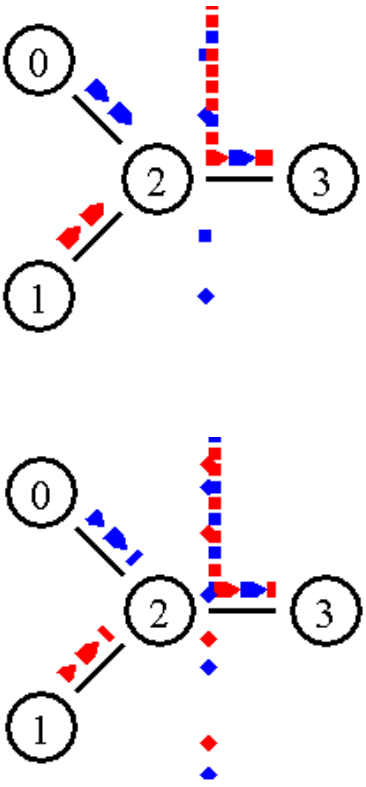
Cuando le damos a play vemos que cado flujo es de un color distinto, pero si observamos la línea del nodo2 al nodo3, vermos que la distribución de tiempo entre los paquetes azules y rojos no es la misma, circulan más paquetes de un color que del otro.

Monitorizar las colas.

Deberemos agregar las siguientes líneas al código que ya tenemos para la línea de n2 a n3

$ns duplex-link-op $n2 $n3 queuePOs 0.5

Repitamos la operación de simular y veremos un grafico similar a este, obsevamos que la cola de paquetes que se genera en n2 solamente caen los paquetes azules. Como una cola en principio no debe ser parcial, lo solucionaremos creando una cola de SFQ para que la línea de n2 a n3 circulen la misma cantidad de paquetes azules y rojos.



Vemos que caen el mismo nº de paquetes rojos que azules

```
#Creamos el objeto a simulador
set ns [new Simulator]

#Define los diferentes colores del flujo de datos
$ns color 1 Blue
$ns color 2 Red
#Abrimos el archivo de trazado
set nf [open out.nam w]
$ns namtrace-all $nf
#Defimos el procedimiento 'finish'
proc finish {} {
	global ns nf
	$ns flush-trace
		#Cerramos el archivo de trazado
	close $nf
		#Execumos en NAM el archivo de trazado
	exec nam out.nam &
	exit 0
}

#Creamos los cuatro nodos
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
#Creamos la union de los distintos nodos
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms SFQ
#Le indicamos donde queremos que aparezcan en NAM los nodos
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

#Monitor de cola para la union entre el nodo 2 y el nodo 3
$ns duplex-link-op $n2 $n3 queuePos 0.5

#Creamos el agente UDP asociado al nodo n0
set udp0 [new Agent/UDP]
$udp0 set class_ 1
$ns attach-agent $n0 $udp0

#Creamos un tr fico CBR de la fuente asociado a udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#Creamos el agente UDP asociado con el nodo n1
set udp1 [new Agent/UDP]
$udp1 set class_ 2
$ns attach-agent $n1 $udp1

#Crear un trafico CBR asociado a la fuante de udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1
#crear un agente nulo (un trafico sink?) asociado al nodo n3
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
#Conectar las fuentes  de trafico con el alsink
$ns connect $udp0 $null0
$ns connect $udp1 $null0
#Planificamos la secuencia de sucesos de los agentes CBR
$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"
#Llamamos al procedimiento finish despues de 5 segundos de simulacion
$ns at 5.0 "finish"
#Hacemos correr la simulacion
$ns run
```

ANEXO B.

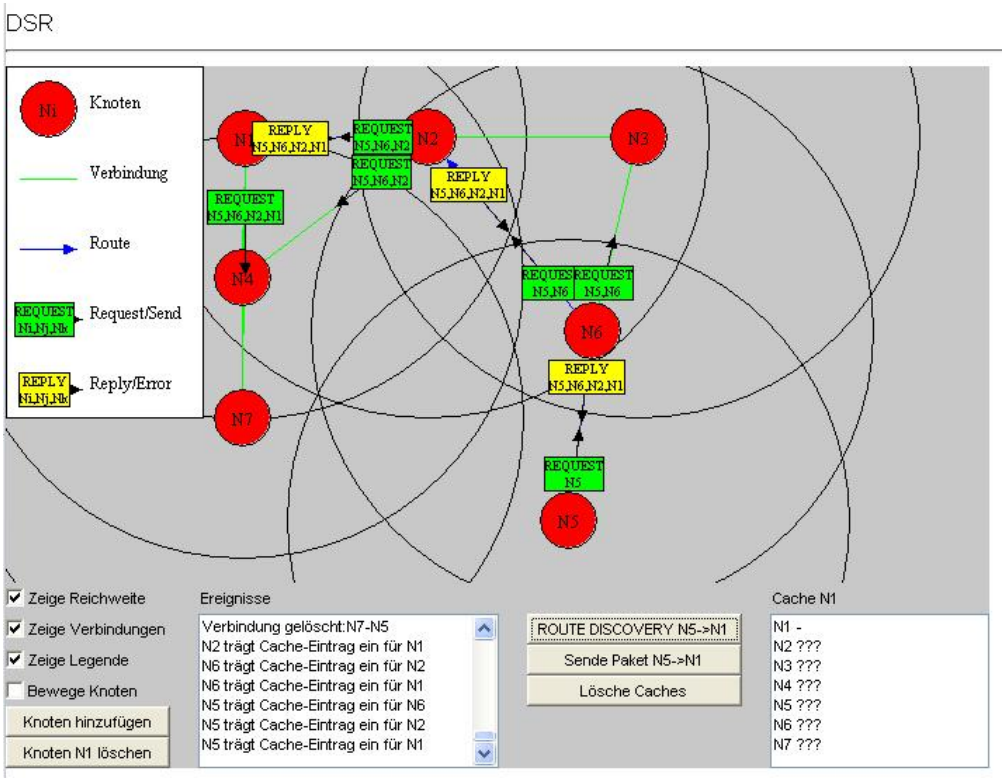
B) Applets para simulación de protocolos inalámbricas.

Gracias a la Universidad de Ciencias Aplicadas de Nuernberg, Alemania, hemos conseguido simular y comprender de una forma muy eficaz el funcionamiento de los distintos protocolos. En su página web se pueden encontrar distintos "*applets"* para simular distintos protocolos, entre ellos el OLSR y el DSR. A continuación vamos a poner algunas capturas de pantalla y explicar brevemente el funcionamiento del mismo. Nótese que en los sucesivos apartados damos por comprendido el funcionamiento de cada protocolo, dichas explicaciones se encuentran en páginas anteriores de este documento.

B.1) Applet DSR

En la siguiente captura observamos una serie nodos que forman una red. Cada uno de los nodos se puede seleccionar y desplazar a lo largo de la imagen recuadrada con fondo gris. Si se desean añadir más nodos se puede presionar sobre el botón con el rotulo "*Knoten hinzufügen"* y se puede borrar un nodo seleccionado con el botón que está justamente debajo de éste con el rotulo "*Knoten x Löschen*" donde x es el nombre del nodo seleccionado.

Justo encima de los botones encontramos una serie de "*check box".* El de más arriba sirve para pintar o no el área de alcance de retransmisión de un mensaje. El segundo sirve para pintar las conexiones entre dos nodos, mostradas en el dibujo con una línea verde. La tercera línea sirve para mostrar o no la leyenda, que se sitúa en la parte superior izquierda de la imagen. Finalmente, la cuarta línea sirve para iniciar o detener una simulación de movimiento de los nodos.



En la parte inferior izquierda aparecen otros 3 botones específicos para éste protocolo. El primero de ellos es para iniciar el descubrimiento de ruta entre un nodo origen y un nodo destino. Para que funcione correctamente, el usuario primero debe de seleccionar dos nodos, X e Y, posteriormente presionar sobre este botón. Así se hará el descubrimiento de la ruta desde el nodo X hasta el nodo Y. En verde se pintan los distintos paquetes de para la etapa de "*Route Request*" . En amarillo, se muestran los paquetes de la etapa de "*Route Reply*". En el cuadrado blanco de la izquierda de estos botones se muestra la traza de envío de los distintos paquetes que se envían para el descubrimiento de la ruta entre origen y destino.

En el cuadrado blanco de la parte inferior derecha se muestra el contenido de la cache de rutas para un nodo seleccionado. Más exactamente, si seleccionamos el nodo N1, en cada fila de este cuadro de texto nos vienen todos los nodos de la red y la ruta para alcanzarlos desde el nodo N1, siempre que la ruta haya sido "escuchada" o procesada por el nodo N1.

El segundo botón, con el rótulo "*Sente Paket X → Y*" es para simular el envío de un paquete desde el nodo X hasta el nodo Y. Este botón funciona de forma análoga al anterior, es decir, es necesario seleccionar dos nodos y luego presionar sobre el botón. El tercer botón, "*Lösche Caches*" sirve para borrar las caches de los nodos de la red. Nótese, que el botón de envío de paquetes el simulador lo realiza con la información que posee en su cache, y no tiene porque ser necesario hacer un descubrimiento de ruta para luego realizar el envío, puede darse el caso de que ya tuviera almacenada la ruta en su cache, de ahí la importancia de las mismas en estos protocolos.

B.2) Applet OLSR

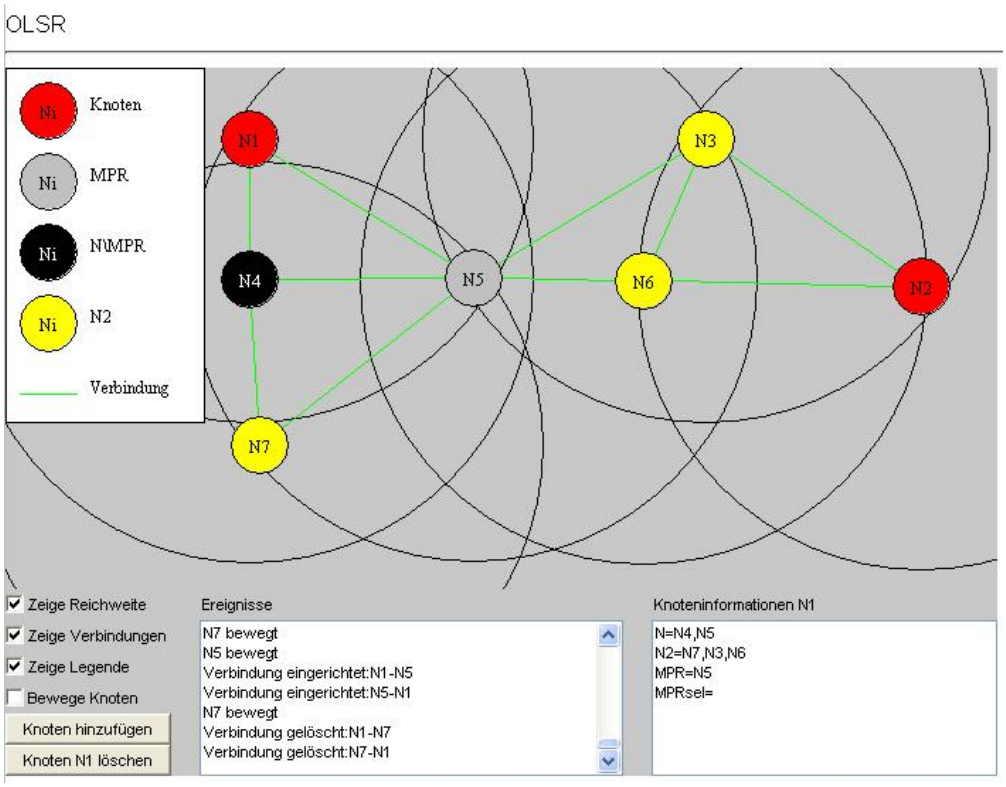
En la siguiente captura observamos una serie nodos que forman una red. Cada uno de los nodos se puede seleccionar y desplazar a lo largo de la imagen recuadrada con fondo gris. Si se desean añadir más nodos se puede presionar sobre el botón con el rotulo "*Knoten hinzufügen"* y se puede borrar un nodo seleccionado con el botón que está justamente debajo de éste con el rotulo "*Knoten x Löschen*" donde x es el nombre del nodo seleccionado.

Justo encima de los botones encontramos una serie de "*check box".* El de más arriba sirve para pintar o no el área de alcance de retransmisión de un mensaje. El segundo sirve para pintar las conexiones entre dos nodos, mostradas en el dibujo con una línea verde. La tercera línea sirve para mostrar o no la leyenda, que se sitúa en la parte superior izquierda de la imagen. Finalmente, la cuarta línea sirve para iniciar o detener una simulación de movimiento de los nodos.

En la siguiente imagen hemos puesto un ejemplo de una posible configuración, el uso en este caso es, si cabe, aún más sencillo que en el del applet del DSR. En este caso, podemos comprobar seleccionando un nodo como cambian los colores de los nodos de su alrededor, siguiendo las siguientes pautas. Para comprenderlo mejor, aclaramos que en el caso de la imagen que hemos añadido hemos seleccionado el nodo N1 de arriba a la izquierda, y que N1 y N7 (abajo a la izquierda) no tienen una conexión directa entre ellos.

Al seleccionar un nodo, se colorean de color gris el conjunto de nodos MPR para el nodo seleccionado, en el caso de la imagen N5. De color negro se pintan aquellos nodos que son vecinos a un salto del nodo seleccionado, en este caso solamente es N4. De amarillo se colorean aquellos nodos que están a 2 saltos del nodo seleccionado, se alcanzan dichos nodos, como sabemos a través de los MPR. El resto de nodos de la red se colorean de color rojo.

En los dos cuadros de textos de abajo se muestra, respectivamente, la información concerniente al nodo seleccionado y la traza de paquetes de control enviados para adquirir dicha información.


OLSR
Ni
Knoten
Ni
MPR
Ni
N\MPR
Ni
N2
Verbindung
N1
N3
N4
N5
N6
N2
N7
Zeige Reichweite
Zeige Verbindungen
Zeige Legende
Bewege Knoten
Knoten hinzufügen
Knoten N1 löschen
Ereignisse
N7 bewegt
N5 bewegt
Verbindung eingerichtet:N1-N5
Verbindung eingerichtet:N5-N1
N7 bewegt
Verbindung gelöscht:N1-N7
Verbindung gelöscht:N7-N1
Knoteninformationen N1
N=N4,N5
N2=N7,N3,N6
MPR=N5
MPRsel=

REFERENCIAS Y BIBLIOGRAFÍA

Gómez C., Paradells J., "Redes ad-hoc: el próximo reto", Buran nº 21, págs. 30 a 37. Mayo 2004.

Kahn R. Et al., "Advances in Packet Radio Technology", Proceedings of the IEEE66: 1468-1496. Noviembre 1978.

Kushalnagar N., Montenegro, G., "6LoWPAN: Overview, Assumptions, Problem Statement and Goals", Febrero 2006.

Web IPv6 over Low power WPAN (6lowpan), URL
http://www.ietf.org/html.charters/6lowpan-charter.html

Kushalnagar N., Montenegro, G., "Transmission of IPv6 Packets over IEEE 802.15.4 WPAN Networks", Octubre 2005.

Plataforma Micaz, URL
http://www.xbow.com/Products/productsdetails.aspx?sid=105

IEEE 802.15.4 standard, URL
http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf

Web grupo MANET, URL http:// www.ietf.org/html.charters/manetcharter.html.

Clausen T., Jacquet P., RFC 3626 "Optimized Link State Routing Protocol (OLSR)". Octubre 2003.

Ogier R., Templin F., Lewis M., RFC 3684 " Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)". Febrero 2004.

Draft OLSR-V2

Perkins C., Belding-Royer E., Das S., RFC 3561 "3561 Ad hoc On- Demand Distance Vector (AODV) Routing". Julio 2003.

Jonson D., Maltz D., Hu Y., "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)" IETF Internet-Draft draft-ietf-manet-dsr-10.txt. Julio 2004.

Chakeres I., Perkins C., "Dynamic MANET On-demand (DYMO) Routing" IETF Internet-Draft draft-ietf-manet-dymo-01.txt. Marzo 2006.

Macker J., "Simplified Multicast Forwarding for MANET", IETF Internet-Draft draft-ietf-manet-smf-01.txt. Marzo 2006.

Kim K., Montenegro G., Daniel Park S., Chakeres I., Yoo S., "Dynamic MANET On-demand for 6LoWPAN (DYMO-low) Routing", IETF Internet-Draft draft-montenegro-6lowpan-dymo-low-routing-00.txt. Octubre 2005.

Kim K., Montenegro G., Daniel Park S., Yoo S., Kushalnagar N., “6LoWPAN Ad Hoc On-Demand Distance Vector Routing (LOAD)”, IETF Internet-Draft draft-daniel-6lowpan-load-adhoc-routing-02.txt. Marzo 2006.

TinyAODV Implementation, TinyOS Source Code Repository http://cvs.sourceforge.net/viewcvs.py/tinyos/tinyos-1.x/contrib/hsn/.

Perkins C., Belding-Royer E., Chakeres I., “Ad Hoc On-Demand (AODV) Routing, IETF Internet-Draft draft-perkins-manet-aodvbis-01.txt. Enero 2004.

Chakeres, I., Klein-Berndt L., "AODVjr, AODV Simplified", ACM SIGMOBILE Mobile Computing and Communications Review pp. 100- 101, Julio 2002.
Web NST-AODV, URL
http://www.i2cat.net/i2cat/servlet/I2CAT.MainServlet?seccio=6_5_2

C. Gomez, P. Salvatella, O. Alonso, J. Paradells, 'Adapting AODV for IEEE 802.15.4 Mesh Sensor Networks: Theoretical Discussion and Performance Evaluation in a Real Environment', 7th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WOWMOM 2006), Niagara Falls, USA. Junio 2006.

Akyildiz I., Wang X., Wang W., “Wireless Mesh Networks: a survey”, Computer Networks Journal (Elsevier), 47(4), 2005.

Chen B., Muniswamy-Reddy K., and Welsh M., “Lessons Learned from Implementing Ad-Hoc Multicast Routing in Sensor Networks”, Harvard University Technical Report TR-22-05, Noviembre 2005.

Draves R., Padhye J., Zill B., “Comparison of routing metrics for static multi-hop wireless networks”, ACM SIGCOMM Computer Communication Review, v.34 n.4, Octubre 2004.

Adya A., Bahl P., Padhye J., Wolman A., and Zhou L., “A multi-radio uni.cation protocol for IEEE 802.11 wireless networks”. In BroadNets, 2004.

De Couto D., Aguayo D., Bicket J., and Morris R., “High-throughput path metric for multi-hop wireless routing”. In *MOBICOM*, Septiembre 2003.

Hu Y. and Jonson D. B., “Design and demonstration of live audio and video over multi-hop wireless networks”. In MILCOM, 2002.

C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc On-Demand Distance Vector (AODV) routing. Internet Request for Comments RFC 3561, Nov. 2003. http://moment.cs.ucsb.edu/pub/rfc3561.txt

M. Guerrero Zapata. Secure Ad Hoc On-Demand Distance Vector (saodv) routing, Sep. 2006. INTERNET-DRAFT draft-guerrero-manet-saodv-06.txt, http://tools.ietf.org/id/draft-guerrero-manet-saodv-06.txt

C. Siva Ram Murthy, B.S. Manoj. AdHoc Wireless Networks. Archiectures and Protocols. Prentice Hall, 2004. ISBN 0-13-147023-X.
Mohammad Ilyas. The Handbook of Ad Hoc Wireless Networks. CRC Press, 2003. ISBN 0-8493-1332-5.

M. Guerrero Zapata. Secure Ad hoc On-Demand Distance Vector Routing. ACM Mobile Computing and Communications Review (MC2R), 6(3):106–107, July 2002.
http://portal.acm.org/ft_gateway.cfm?id=581312&type=pdf

G. Montenegro and C. Castelluccia. Statistically unique and cryptographically verifiable (SUCV) identifiers and addresses. Network and Distributed System Security Symposium (NDSS ’02), Feb. 2002.
http://www.isoc.org/isoc/conferences/ndss/02/proceedings/papers/monten.pdf

http://www.wikipedia.org

M. Guerrero Zapata and N. Asokan. Securing Ad hoc Routing Protocols. In Proceedings of the 2002 ACM Workshop on Wireless Security (WiSe 2002), pages 1–10, September 2002.
http://www.cs.huji.ac.il/labs/danss/sensor/adhoc/routing/zapta_2002securingadhocrouting.pdf

Huaizhi Li and Mukesh Singhal. A Secure Routing Protocol for Wireless Ad Hoc Networks. In Proceedings of the 39th Hawaii International Conference on System Sciences, pages 225.1, January 2006.
http://doi.ieeecomputersociety.org/10.1109/HICSS.2006.29

M. Guerrero Zapata. Securing and Enhancing Routing Protocols for Mobile Ad hoc Networks. Doctoral Thesis. Computer Architecture Department. University of Catalonia, 2006.
http://www.tdx.cbuc.es/TDX-1212106-103102/

P. Papadimitratos and Z. J. Haas. Secure routing for mobile ad hoc networks. SCS Communication Networks and Distributed Systems Model-ing and Simulation Conference (CNDS 2002), Jan 2002.
http://citeseer.ist.psu.edu/rd/9909163%2C559572%2C1%2C0.25%2CDownload/http://coblitz.codeen.org:3125/citeseer.ist.psu.edu/cache/papers/cs/27088/http:zSzzSzwnl.ece.cornell.eduzSzPublicationszSzmc2r_10_02.pdf/papadimitratos02secure.pdf

Y. Lin, A. Hamed Mohsenian Rad, Vincent W. S. Wong, Joo-Han Song and Joo-Han song. Experimental comparisons between SAODV and AODV routing protocols. In International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems and Proceedings of the 1st ACM workshop on Wireless multimedia networking and performance modeling, pages 113-122, October 2005.
http://portal.acm.org/citation.cfm?id=1089757

M. Guerrero Zapata. Simple ad hoc key management (sakm), Feb. 2006. INTERNET-DRAFT draft-guerrero-manet-sakm-00.txt
http://tools.ietf.org/id/draft-guerrero-manet-sakm-00.txt

C. Madson and R. Glenn. The use of HMAC-MD5-96 within ESP and AH. Internet Request for Comments RFC 2403, Nov. 1998.
http://rfc.net/rfc2403.html

C. Madson and R. Glenn. The use of HMAC-SHA-1-96 within ESP and AH. Internet Request for Comments RFC 2404, Nov. 1998.
http://rfc.net/rfc2404.html